Juho Knuuttila

Automatic self-supervised learning of associations between speech and text

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 9.3.2015

Thesis supervisor:

Prof. Unto K. Laine

Thesis advisor:

D.Sc. (Tech.) Okko Räsänen

Aalto University
School of Electrical Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Juho Knuuttila

Title:
Automatic self-supervised learning of associations between speech and text

Date: 9.3.2015 **Pages:** x + 53

Professorship: Acoustics and Audio Signal Processing **Code:** S-89

Supervisor: Prof. Unto K. Laine

Advisor: D.Sc. (Tech.) Okko Räsänen

One of the key challenges in artificial cognitive systems is to develop effective algorithms that learn without human supervision to understand qualitatively different realisations of the same abstraction and therefore also acquire an ability to transcribe a sensory data stream to completely different modality. This is also true in the so-called Big Data problem. Through learning of associations between multiple types of data of the same phenomenon, it is possible to capture hidden dynamics that govern processes that yielded the measured data.

In this thesis, a methodological framework for automatic discovery of statistical associations between two qualitatively different data streams is proposed. The simulations are run on a noisy, high bit-rate, sensory signal (speech) and temporally discrete categorical data (text). In order to distinguish the approach from traditional automatic speech recognition systems, it does not utilize any phonetic or linguistic knowledge in the recognition. It merely learns statistically sound units of speech and text and their mutual mappings in an unsupervised manner. The experiments on child directed speech with limited vocabulary show that, after a period of learning, the method acquires a promising ability to transcribe continuous speech to its textual representation.

Keywords: statistical learning, associative learning, weakly supervised learning, self-supervised learning, pattern recognition, machine learning

Language: English

AALTO-YLIOPISTO
SÄHKÖTEKNIIKAN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä:	Juho Knuuttila		
Työn nimi:	Puheen ja tekstin välisen tilastollisen assosiaation itseohjautuva oppiminen		
Päiväys:	9.3.2015	**Sivumäärä:**	x + 53
Professuuri:	Akustiikka ja äänenkäsittelytekniikka	**Koodi:**	S-89
Työn valvoja:	Prof. Unto K. Laine		
Työn ohjaaja:	TkT Okko Räsänen		

Keinoälyn toteuttamisessa vaikeimpia haasteita on kehittää ohjaamattomia oppimismenetelmiä, jotka oppivat yhdistämään saman abstraktin käsitteen toteutuksen useassa eri modaaliteeteissa ja vieläpä kuvailemaan aistihavainnon jossain toisessa modaaliteetissa, missä havainto tapahtuu. Vastaava pätee myös niin kutsutun Big Data ongelman yhteydessä. Samasta ilmiöstä voi usein saada monimuotoista mittaustuloksia. Selvittämällä näiden tietovirtojen keskinäiset yhteydet voidaan mahdollisesti oppia ymmärtämään ilmiön taustalla olevia prosesseja ja piilevää dynamiikkaa.

Tässä diplomityössä esitellään menetelmällinen tapa löytää automaattisesti tilastolliset yhteydet kahden ominaisuuksiltaan erilaisen tietovirran välille. Menetelmää simuloidaan kohinaisella sekä korkea bittinopeuksisella aistihavaintosignaalilla (puheella) ja ajallisesti diskreetillä kategorisella datalla (tekstillä). Erotuksena perinteisiin automaattisiin puheentunnistusmenetelmiin esitetty menetelmä ei hyödynnä tunnistuksessa lainkaan foneettista tai kielitieteellistä tietämystä. Menetelmä ainoastaan oppii ohjaamattomasti tilastollisesti vahvat osaset puheesta ja tekstistä sekä niiden väliset yhteydet. Kokeet pikkulapselle suunnatulla, sanastollisesti rajoitetulla puheella osoitti, että oppimisjakson jälkeen menetelmällä saavutetaan lupaava kyky muuntaa puhetta tekstiksi.

Asiasanat:	tilastollinen oppiminen, assosiatiivinen oppiminen, heikosti ohjattu oppiminen, itse-ohjautuva oppiminen, hahmontunnistus, koneoppiminen
Kieli:	Englanti

Acknowledgements

All the research for this thesis was conducted in the Department of Signal Processing and Acoustics, Aalto University.

I would like to thank my supervisor professor Unto K. Laine and advisor Okko Räsänen for the opportunity to work with this intriguing research topic. It was a privilege to work in such an inspiring working environment. I am truly grateful to Unto and Okko for all the guidance and their patience to take the time to support me, despite my moderate pace of writing. Thanks are also extended to Jan Hoffman for proof-reading.

I also want to thank my family for their support and my friends—especially the fellows at "the party engineers"—for making my time as a student memorable.

Helsinki, 19.1.2015

Juho Knuuttila

Contents

List of Tables

List of Figures

Abbreviations and acronyms

ASR	automatic speech recognition
CFG	context free grammar
CM	concept matrix
DCT	discrete cosine transformation
DFT	discrete Fourier transformation
DP-ngrams	dynamic programming ngrams
DTW	dynamic time warping
EM	expectation-maximization
FFT	fast Fourier transformation
HHMM	hierarchical hidden Markov model
HML	hybrid model learner
HMM	hidden Markov model
IG	information gain
MDL	minimum description length
MFCC	Mel-frequency ceptsral coefficient
MI	mutual information
NMF	non-negative matrix factorization
RED	relative edit distance
SCFG	synchronous context free-grammar
SED	sum of edit distances
SLCM	self-learning concept matrices
SMT	statistical machine translation
STT	speech-to-text
VQ	vector quantized

Chapter 1

Introduction

Conceptual learning in the human cognitive system never occurs inside a single modality, but in terms of associations between representations in multiple perceptual modalities and motor outputs. As the events in our environment often provide information through multiple modalities, learning can also occur through co-occurrences of structured activities at different modal dimensions. In this context, pattern discovery in one modality is basically only data segmentation or clustering and the created clusters are meaningless without grounding through *multimodal associations*. [tBC07] [YBA05] [MD09]

Furthermore, the same principles can be applied to the so called Big Data problem in modern society. Discovery of statistically significant patterns from data and learning of associative links between qualitatively different data streams can bring more understanding of the underlying principles governing the processes. These can be anything from industrial manufacturing and fault detection to customer behavior.

The main purpose of this thesis, at a highly conceptual level, is to develop general methodology to analyze, discover, and model associations between two qualitatively different sequential data streams. The problem is modeled as an unsupervised pattern discovery problem which is converted into a *self-supervised* learning process. Here, the self-supervised learning process involves first automatically deriving representations in one modality and and then using them to aid discovery of patterns in an another modality.

The term modality generally refers to human physiology and sensation, but in machine learning discipline it could basically be any measurement data. The two "modalities" used in the current work are spoken English utterances and textual representations corresponding to the utterance contents but with all white spaces and special characters removed. The removal of these characters results a representation that mimics continuous speech. It

is practically impossible to recognize individual words from a totally unfamiliar or the very first language due to absence of acoustic hints for word boundaries.

Speech and text are chosen as the data types for this study due to the fact that despite their strong mutual interdependency, they are clearly qualitatively different, have different temporal characteristics, but the results of the associative pattern discovery are still easy to evaluate. Most importantly, the basic units in text (i.e., letters) do not have direct correspondence to any units of speech that could be defined purely based on the raw acoustic signal and the pronunciation of the letters depends on the lexical context so that each letter becomes realized in various acoustic forms. For example, consider the pronunciation of the letter *c* in the words *ocean* and *cat*.

In general, one can hypothesize that the statistical connection between structure of speech and text is most significant at the level of word-like patterns of speech and text instead of the low-level feature/letter representations of the modalities. In order to learn the dependencies between the modalities, one has also to learn these temporally spanning patterns first. Although only text and speech are used in this study, the same proposed concept should be applicable also to other pairs of sequential data streams that model the same phenomena or are hypothesized to have high correlation for some other reason.

The quality of the current approach is evaluated as its capability to transcribe speech to text and it resembles an automatic speech recognition (ASR) or a speech-to-text (STT) system [RJ93]. However, creating such a system is not the goal per se, but only a way to prove the concept. It is in contrast to the ASR systems that rely heavily on linguistics and on recognition of phonemes or their combinations. These linguistically motivated units are recognized mostly without any semantic component or associations to other modalities or information sources. To make a clear difference to such systems, it should be emphasized that the approach that will be presented in this thesis is based merely on the signal statistics within and between the two different data streams, and not based on any prior phonetic or linguistic expert knowledge. Similarly, a child learning multimodal association does not have this kind of expert knowledge and is still able to learn to speak and to understand spoken messages. Some domain expertise is used only in the selection of suitable machine learning algorithms for every given sub-problem.

The basic objective of this thesis is to solve the generic problem of finding the statistical relationship between two low-level unlabeled asynchronous signals. In practice it is a task of maximizing the predictability of one signal when the other is given and in the end optimizing the parameters of algorithms chosen for the methodology. As shall be seen, there are a lot of

parameters that could be fine tuned and many more could be applied. The proposed methodology is not claimed to be optimal as far as the parameters are considered, but rather sub-optimal when some of the parameters are fixed. The aim is to give a proof that the methodology is sensible, robust and potentially beneficial.

The structure of this thesis is as follows: first, in the next chapter the concepts of machine learning and pattern recognition that are important in terms of this thesis are presented along with a brief literature review. Additionally, the terminology is defined. It is important because the machine learning as a discipline itself is juvenile compared to basic sciences and therefore the terminology is wide and very variable. There are various possible ways and terms to describe a given methodology , and inversely, some of the terms might mean different things to different people. The literature review remains short because similar studies are hard to find due to freshness of the given approach and due to already mentioned variability in terminology. Second, in chapter 3 the *associative learner* algorithm is presented. Then, in chapter 4 the experiments and the results are presented. Finally, chapter 5 concludes this thesis with discussion of the results and potential future work.

Chapter 2

Background

This chapter presents the theoretical aspects and basics of machine learning in the scope that is required to understand the self-supervised machine learning methodology that will be presented later in this thesis. These aspects include, among others, the general process of pattern discovery and recognition. Examples of every step of the process are given, emphasizing mostly on methods used to build cross-modal associative learner presented in chapter 3. Along the way the terminology is fixed since it is not universally agreed inside the machine learning discipline. Additionally, some specific applications are reviewed that are either relevant to the general research area or semantically close to machine learning problem presented in this thesis.

The terms machine learning and pattern recognition refer to algorithms that learn from data. Generally the goal of a machine learning algorithm is to either learn to recognize and label previously unseen data or predict upcoming values of measurements. The recognition or prediction ability is acquired through making generalizations of seen data. Once the model is generated, recognition and labeling of patterns from previously unseen data is possible. [Bis06]

A pattern is a discernible regularity in the data. In other words, the elements in the pattern repeat in a predictable manner. The patterns having mutually resembling regularity belong to the same pattern class and are given a common class label. A model for a pattern class describes the regularity it in such a way that it captures the statistical variability of the patterns belonging under the same class label and discriminates them from other patterns belonging under different labels.

The generalization capability of a model is important. With some learning algorithms it is possible to train a classifier so that it reaches very high, or even 100%, recognition accuracy in the training set, but fails miserably in classification of new, unseen data. This is called *overtraining* or *overfitting*

[Bis06], in which the classifier is trained to recognize patterns that are too similar to those noisy exemplars found in the training set.

The learning, or generation of the models, can be categorized in to *batch* and *online* modes according how the data is processed and stored. In batch processing, the models are first created from a training dataset and then validated with another, usually disjointed dataset. Some learning algorithms working in batch mode involve iteration to fit the model parameters [SW96] and therefore storing or "remembering" all the data is required, while others [RLA09] do single pass execution on the data, after which the data can be forgotten. In online learning the models are created and updated on the go, as new data arrive.

Machine learning algorithms are categorized among the contents and the quality of the training data. At the opposite ends are supervised and unsupervised learning according to the availability of labeling. The term weakly supervised learning in classification problem is used in cases where the exemplars are not exact or the labeling is noisy. In this thesis it is used to denote a situation when a learner is given a label and a training vector (actually a sequence of symbols), that is known to contain the corresponding pattern and some extra irrelevant information. The training vector could be significantly longer than the actual pattern and therefore the whereabouts (the first and the last indices) of the actual pattern (or sub-sequence) is unknown. Examples of each types of learning algorithms are given later in this chapter.

There are machine learning methods for many types of data, ordered and unordered, annotated and un-annotated. In this study order of data is significant. Sequence is formally an ordered list of symbols or objects, for example notes, characters or nucleic acid sequence in DNA. Changing the order of any of them in a sequence results in a different melody, word or genome.

A time-series is a sequence of data points usually resulting from sampling or measuring a random variable at uniform time intervals. The measured random variable can be anything from sound pressure, temperature, and stock value to body weight. A multivariate time-series is high dimensional version of uni-variate time-series: it is a sequence of vectors, or a matrix if you will.

A general pattern recognition process is illustrated in figure 2.1. The pattern recognition is a process of assigning label(s) to input, which might be a single value, a sequence or a time-series. The labels are references to patterns. The recognition process involves following steps:

1. (possible) preprocessing

2. feature extraction

3. classification

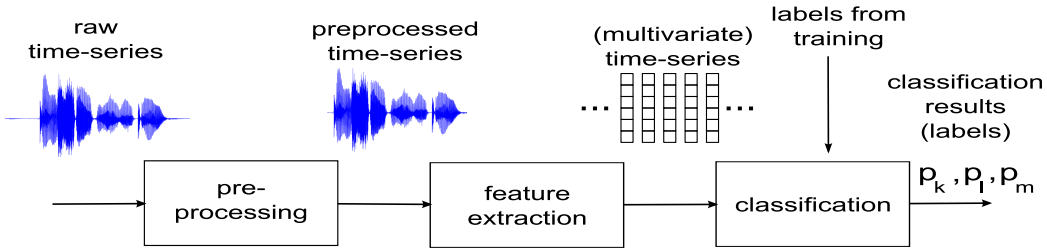


Figure 2.1: *Block diagram of pattern recognition*

In figure 2.1 the input is speech audio. The speech audio is a uni-variate time-series representing perceived instantaneous sound pressure. The labels representing patterns could be anything: textual representations of the speech, speaker identification, or emotion recognition. First, the input signal might be preprocessed for easier handling of the signal. The prepocessing can involve something like down or up sampling, segmenting etc. Then, selected features are extracted from the preprocessed signal. Usually, at least at some point, multiple features are extracted from the signal. The extraction thus produces a series of feature vectors, which can be continuous valued or symbolic in nature. The multivariate feature vectors could also be processed further depending on the classifier. Finally, the classifier makes classification decisions based on a prebuilt model and assigns labels to feature vectors.

The patterns are here defined to be top level concepts that might even be called abstractions. Once created or discovered they are constant. Depending on data, instances of a single pattern might be variable. At the level of raw data these instances of the same pattern can differ in content and in length. Considering speech audio, words, phrases and utterances sound different between different speakers (e.g. male and female speakers). The variance can even be high with a single speaker. The prosody changes whith, for example, the emotional state or health (sarcasm or flu) of the speaker. Actually, the waveform of the speech is practically never exactly the same even with a single speaker in a controlled environment. For these reasons the raw data by itself is not good for discovery and recognition purposes. In this case the recognition is based on some structure or features extracted from the raw data.

Before pattern recognition is possible, a classifier has to be build and adequate features for the classifier selected.

The case where desired output values and corresponding exact exemplars are given to the algorithm is called supervised learning. If the input and output values are discrete or symbolic, they are also called labels. The output of the training is a function called a classifier or regression for discrete and continuous valued output respectively. The function should be able to produce correct output values for input data.

At the opposite end of this classification of learning algorithms is unsupervised learning. The problem is quite different since there are no desired output values or labels to pass to the algorithm. Instead of training with labels the task is to find structure from unlabeled data. The lack of labeling makes the evaluation of learning results harder than in supervised algorithms. There is no straightforward way of determining the validity of the inferred structure, unless labeling exists for development purposes.

The classification of a machine learning method to supervised or unsupervised is generally also perceived to refer to the amount of human intervention and manual labor needed. The unsupervised methods are often perceived as automatic. All machine learning algorithms cannot be classified strictly to either of these two classes. Some of the algorithms fall somewhere in between. These kind of methods could be referred as semi-automatic when hand-editing is done along with the training or as weakly supervised when the training exemplars are not exact instances of jointly given labels.

2.1 Feature extraction

A very important part of machine learning application design is feature extraction. Feature extraction involves the selection of the quality and the quantity of the features. Since there is no general theory that would indicate the optimal set of features, the selection is based mostly on varying combination of domain expertise, intuition, and trial-and-error. Feature selection is such a complex problem that it is an active research topic alone. The problem of feature selection is reviewed for example in [BL97].

The features depend on the application and purpose of the machine learning algorithm. The features should carry relevant information about the data they were extracted from in order to make the desired task easier, therefore the overall quality of a classifier is the result of a good match between appropriate features and the classifier.

Usually the amount of available training data limits the feasible number of features. In the cases where there is too little data compared to the amount of training data, all the data appears to be sparse. This leads also to high risk of over fitting to the training data. Pattern discovery often relies on

finding groups of similar feature vectors. This means that the developed model describes the training data, but makes poor generalization to unseen, future data, thus the predictive power of the machine learning algorithm is reduced. Therefore the amount of training needed to get reliable results grows exponentially along the dimensionality of features. This phenomenon is called the curse of dimensionality [Bel72].

It is often impractical or impossible to operate with and recognize the pure raw data itself due to its variability. The expressions of the same pattern might look different when looking only at the raw data or in the case of speech its acoustic waveform. Some extractable and descriptive features of the data are therefore more distinguishable. Speech is known to have high variability already with one single speaker not to mention between two speakers. The problem is to extract features that can capture variant expressions of constant models of a pattern.

2.1.1 MFCC

One of the most used features in ASR is Mel-frequency ceptsral coefficients (MFCCs) [RS07]. It was originally formulated in [DM80]. MFCCs are used as features for example in speaker recognition [GFK05] and music genre classification [MÖ07]. Speech can be considered a piecewise quasi-stationary process, so reliable spectral information of small time frames can be used in feature extraction for pattern discovery and recognition. This is also the idea behind in extraction of MFCCs. They constitute a short term power spectrum of sound.

The features are extracted from short, partially overlapping, excerpts of the series. These excerpts are called windows. The window size in the case of audio is usually around few tens of milliseconds and the step size around ten milliseconds. The step size is also the time interval of the resulting feature vectors.

The process of extracting the MFCC features from a single window of time series is presented in figure 2.2. First, an analysis windowing function is applied in order to dampen the time series values at the edges of the window and discrete Fourier transformation (DFT), or fast Fourier transformation (FFT) in practical applications running on computers, is used to convert the time-series to frequency domain:

$$X_i[k] = \sum_{n=0}^{N-1} y_i[n]h[n]e^{-i2\pi k\frac{n}{N}}, \quad 0 \leq k \leq K-1 \tag{2.1}$$

where $y_i[n]$ is the excerpt of the time-series of N samples and $h[n]$ analysis

window with the same length. K is the length of DFT. X_i is the obtained frequency spectrum of $y_i[n]$. Subindex i just denotes the operation is performed on i:th window of the time-series.

Taking absolute value and raising to the second power each term of the obtained spectrum gives the corresponding power spectrum:

$$P_i[k] = |X_i[k]|^2, \quad 1 \leq k \leq K-1 \tag{2.2}$$

A Mel-scale filter bank is applied to the power spectrum to mimic human hearing properties. It is a collection of M triangular band pass filters, whose center frequencies and bandwidths increase logarithmically. The power in each band is calculated by summing the power spectra over the bands. This results in Mel spectrum MS, a vector of M values.

Finally, taking a logarithm of MS and applying discrete cosine transformation (DCT) results the MFCC feature vectors:

$$MFCC_i[m] = \sum_{m=0}^{M-1} \log_{10}(MS_i[m]) \cos\left[k(m+\frac{1}{2})\frac{\pi}{M}\right], \quad 1 \leq k \leq M-1 \tag{2.3}$$

In the equation above the indexing of k starts from 1, not 0, because the first (DC-)component of DCT is omitted here.

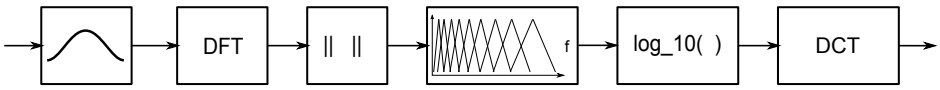


Figure 2.2: *Block diagram of extracting Mel-frequency cepstral coefficients from an excerpt of a time-series. 1. Hamming windowing, 2. Discrete Fourier transformation, 3. Power spectrum, 4. Mel-filtering, 5. Logarithm, 6 Discrete cosine transformation.*

2.2 Unsupervised learning

The objective in unsupervised learning is to find structure from previously unknown unlabeled data. The concept is closely related and partially overlapping to data mining and statistics due to fact that it is applied to many types of data and applications like: finding market segments from customer databases (facebook, google, retail marketers), word discovery from continuous speech, grammatical inference and so on.

The unsupervised learning is called *dimensionality reduction* when high dimensional data is projected to lower dimension for visualisation purposes or

to find the most relevant features. Examples of such techniques are principal component analysis [Hot33], diffusion maps [CL06] and random projection [RK89]. In the case of *density estimation*, the aim is to find how data is distributed. In *clustering*, the target is to form groups of observations, or *clusters*, so that members of a cluster are more similar or related to each other than the rest of the observations.

Unsupervised learning methods can be used in the pre-processing of the data before the actual discovery or training. Such applications of unsupervised learning includes, but not limited to, dimensionality reduction before clustering and/or clustering before training.

A few unsupervised learning methods that are important in this thesis are outlined in the following subsections: a centroid based and a hierarchical/agglomerative clustering methods. An example of both can be seen in figures 2.3 and 2.4.

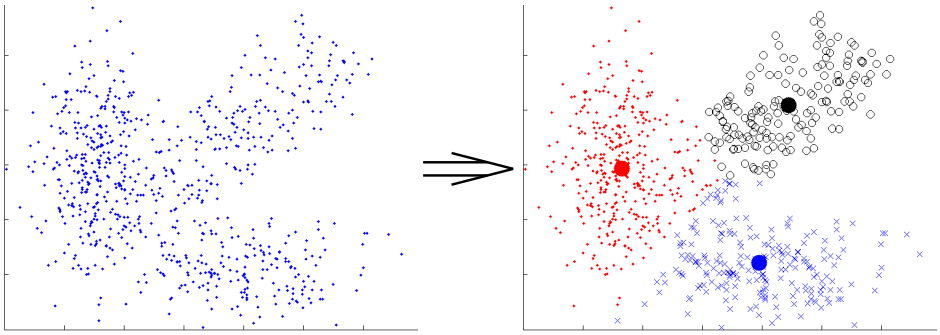


Figure 2.3: *An example of k-means-clustering. In the left plane is a group of datapoints and in the right is a result of k-means clustering with 3 clusters common Euclidean distance as selection criterion. The shape and the color of the markers denote the assignment to each cluster. The large solid dots denote the computational cluster centroids at the end of the iteration of the algorithm (convergence).*

2.2.1 Clustering

Clustering is divided in to hard and soft clustering according to which degree a single observation can belong to a cluster. Each observation either belongs to a cluster or not in the hard clustering as opposed to the soft clustering, where each observation belongs to every cluster with an individual degree. When the number of clusters, to which an observation can belong to, is limited to one, the clustering method is called strictly partitioning clustering.

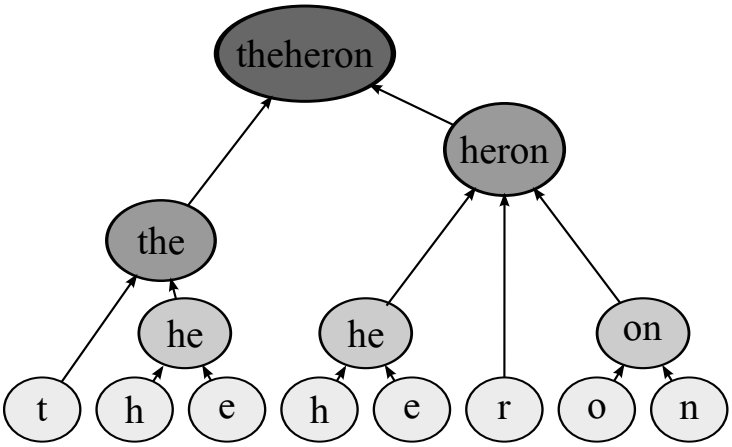


Figure 2.4: *Hierarchical clustering. The clustered data does not necessarily have to be sequential as in this example. It can be composed of practically any ordered or unordered entities like customer segmentation criteria.*

Probably the most widely used hard clustering algorithm for multivariate data is the k-means clustering algorithm. It is hard, strictly partitioning, clustering method. The k-means is based on assigning observations to the nearest mean, called a cluster centroid. It is a simple iterative algorithm with guaranteed convergence, not necessarily to globally optimal but some suboptimal solution [Mac67].

The partitioning a set of observations to k clusters k-means algorithm is iterated as follows:

1. initialize centroids by for example randomly selecting k observations
2. assign every observation to its closest centroid
3. compute the mean of each cluster and move the centroids to corresponding means
4. continue from step 2, unless none of the centroids is moved in step 3 or pre-defined number of iteration rounds is reached

The categorization of future observations is straightforward, assign new observations to their closest centroid.

2.2.2 Grammatical inference

Grammatical inference, also referred as syntactic pattern recognition [Sch92], is a machine learning process that induces a formal grammar from observa-

tions. The formal grammar is usually presented in the form of production or re-write rules.

A grammar for given observations can be induced with several methods falling mainly in to three different categories: trial-and-error [DHS00], genetic algorithms [CK10] and greedy algorithms [Sol64] or their combinations using for example handcrafting to support otherwise automatic greedy algorithm [WMS01].

Here, the focus is on the greedy methods, because in the methodology presented in this thesis the formation of a formal grammar—the source of labeling— is based on greedy agglomeration. The greedy algorithms are data driven, iterative and deterministic. At every iteration the grammar is optimally either extended or modified. The definition of optimality depends on assigned selected objective criterion. Since the idea behind in the proposed methods is to look for statistical dependencies inside and across two sequential data streams that both are known to have temporal dependencies, straightforward greedy inference methods serve the purpose well because they are easy to assign and evaluate a mathematical function as the objective criterion.

In the following examples of inferring a context free grammar (CFG) the iteration means strictly appending of a production rule to the grammar. The selection of the symbol pair for the rule is based on different statistical measures.

A CFG is generally defined by two finite sets of symbols, terminals Σ and non-terminals V, a finite set of production rules R from V to $(\Sigma \cup V)^*$ and a starting symbol S. Here, the production rules include only symbol pairs $V \rightarrow [\Sigma \cup V][\Sigma \cup V]$, meaning replacement of a non-terminal symbol with a pair of symbols, where both symbols can be either terminal or non-terminal independently from each other. A production rule is written in terms of symbols as:

$$\alpha_i \rightarrow a_k a_l, \tag{2.4}$$

where i is the number of iteration and sub-indices k, l refer to position in the sets of terminals and non-terminals, not the observations. Indices below or equal to the cardinality of the set of non-terminals denote that the symbol is a terminal symbol and indices greater than those denote that the symbol is a non-terminal symbol.

2.2.2.1 Iteration of the inference

One iteration round in the inference of a CFG consists of two parts: selection of the symbol pair for the production rule and using the selected rule in the opposite direction, that is compressing the sequence of observations by

agglomeration. The first and the most straightforward idea is to select the symbol pair with the highest number of occurrences $F(a_k, a_l)$, resulting in maximal local compression [Sol64]. However, maximal local compression does not necessarily lead to a grammar that is optimal for pattern recognition purposes. The mutual information (MI), on the other hand, is based on the statistical dependencies between the discrete elements in the data. Mutual information of a symbol pair $a_k a_l$ in the observations is expressed as

$$MI(a_k, a_l) = P(a_k, a_l)\log_2\frac{P(a_k, a_l)}{P(a_k)P(a_l)}. \tag{2.5}$$

In equation 2.5 $P(a_k, a_l)$ is the probability of the symbol pair, where the symbol a_k is followed by the symbol a_l, and $P(a_l)$ and $P(a_k)$ are the probabilities of individual symbols. In place of the probabilities their estimates—frequencies of occurrences—are used. For example for symbol pair a_k, a_l the estimate is defined as $\hat{P}(a_k, a_l) = \frac{F(a_k,a_l)}{N-1}$, where N is the length of the sequence. There are always $N-1$ symbol pairs in N symbol long sequence.

Wu and Su have pointed out that MI is prone to estimation errors especially when the number of occurrences of both symbols are low [WS93]. Wong et al. have proposed that information gain (IG), as presented in equation 2.6, could alleviate this problem. It measures the amount of information in bits about one symbol that is obtained by the knowledge of existence or absence of the other symbol:

$$\begin{aligned} IG(a_k, a_l) = {} & P(a_k, a_l)\log\frac{P(a_k, a_l)}{P(a_k)P(a_l)} + P(a_k, \bar{a}_l)\log\frac{P(a_k, \bar{a}_l)}{P(a_k)P(\bar{a}_l)} \\ & + P(\bar{a}_k, a_l)\log\frac{P(\bar{a}_k, a_l)}{P(\bar{a}_k)P(a_l)} + P(\bar{a}_k, \bar{a}_l)\log\frac{P(\bar{a}_k, \bar{a}_l)}{P(\bar{a}_k)P(\bar{a}_l)} \end{aligned} \tag{2.6}$$

In the equation $P(a_k, \bar{a}_l) = \frac{Freq(a_k)-Freq(a_k,a_l)}{N-1}$ and $P(\bar{a}_k, \bar{a}_l) = 1-P(a_k, \bar{a}_l)-P(\bar{a}_k, a_l)$

Recently Laine has proposed that the inference of the grammar could be driven by entropy rate (h). When recurring structures are removed from observation sequences through agglomeration, the sequences appear more random and the entropy rate increases. Selecting the symbol pair causing the largest Δh for the agglomeration/new production rule is the core idea behind the hybrid model learner (HML) [Lai11].

2.2.2.2 Stopping criteria

To be a truly unsupervised learning mechanism, the inference of grammar or any other iterative algorithm needs an automatic stopping criterion. Again,

there are plenty of choices.

In the case of grammars, their coverage can be used as the stopping criterion. The coverage is here defined as the percentage of terminal symbols in the training data set that are captured in the grammar. For pattern recognition purposes a more natural way is to consider for example the relative growth of the coverage. It is based more on statistics and requires less knowledge about the training set. Setting too high a coverage threshold for stopping criterion without any knowledge about the variability of the training set has a risk of serious overfitting. The relative growth as stopping criterion was used in grammar induction by Wong, et al. [WMS01].

A very simple stopping criterion would be a lower limit for the maximal number of occurrences of any symbol pair in the compressed sequence. Also, when the increase of entropy rate drops it is an indication that there are less recurring structures to be found. In subsection 2.2.2.1 it was implied that the reliability of statistical estimators decreases along the number of occurrences of corresponding symbols. This leads to the idea that the inference could be stopped when this reliability falls too low. The low reliability could indicate that the grammar is starting to overfit in to the training data. Lesne et al. defines the reliability as *good statistics* and put it in mathematical form as condition [LBP09]:

$$\max P(a_i, a_j) N_{eff} \gg 1, \tag{2.7}$$

where N_{eff} is the effective length of the actual length N of the symbol sequence after $|V|$ iterations. It is written open in equation 2.8:

$$N_{eff} = \frac{Nh}{\ln |K|}, \tag{2.8}$$

where N is the actual length of the compressed sequence, h the entropy rate, and K the number of the unique symbols in the compressed sequence.

When grammatical inference is used solely for data compression purposes the stopping criterion is naturally the description length. The description length is the total sum of bits needed to fully describe the original data, i.e. compressed data plus compression rules in bits. The compression (and the inference of the grammar) is stopped when the total amount of bits needed for decoding the compressed sequence and the inferred grammar would start to grow. The minimum description length (MDL) would be the method and the optimal stopping point giving best possible compression [Ris78]. It is a formalization of Occam's razor principle and states that best hypothesis for a given set of data leads to the best possible compression of the data.

2.2.3 Pattern discovery from time series

Previously discussed grammatical methods do not suit directly to a time series without quantization. Park and Glass have proposed a truly unsupervised method for finding patterns from speech. They used their own modification of dynamic time warping (DTW) to find similar sub-sequences from pairs of spoken utterances. The sub-sequences were then clustered by presenting them as adjacency graphs. In their experiments the found speech patterns were shown to correspond to words and phrases relevant to the audio streams they were extracted from. [PG08]

DTW was originally used as a way to compare two spoken instances of a word. As stated earlier these instances are practically never the same. The DTW can be used to alleviate problems especially originating from different talking speeds which mathematically equal to different lengths of the vectors representing the instances. The comparison is based on the accumulated distortion along an optimal alignment of exemplars. Two words, $\mathbf{X}$ and $\mathbf{Y}$, can be presented as time series of feature vectors, for example MFCCs, $(\mathbf{x}_1, \ldots, \mathbf{x}_{N_x})$ and $(\mathbf{y}_1, \ldots, \mathbf{y}_{N_y})$ respectively. The alignment is a mapping from $\mathbf{X}$ to $\mathbf{Y}$ with constraints. Mathematically the alignment, or the warp path ϕ, is a sequence of ordered pairs:

$$\phi = (i_k, j_k) \quad k = 1, \ldots, T \tag{2.9}$$

The globally optimal alignment is the one that minimizes

$$D_\phi = \sum_{k=1}^{T} d(\mathbf{x}_{i_k}, \mathbf{y}_{j_k}), \tag{2.10}$$

where $d(\cdot, \cdot)$ is a distance function between two vectors.

In the most basic DTW for word comparison the start and end points of warp paths are the start and end points of the time series under comparison: $(i_1, j_1) = (1, 1)$ and $(i_k, j_k) = (N_x, N_y)$. The segmental DTW does not include this restriction and allows different starting and ending points, but it has more restrictions for the shape of the warp path. The segmental DTW thus produces a family of warp paths by varying the start and end points. The warp paths are refined by discarding portions with high distortions. Setting an appropriate lower limit for the warp path length meaningful similar segments from an utterance pair can be extracted. The extracted information from the utterance pair is two time intervals and associated distortion measure for the two segments.

From similar segment pairs an adjacency graph that represents distortions (or the similarities) between all the discovered segments is constructed. The

common segments from two pairs are recognized through time alignment. For example if similar segments are found in utterance pairs (A, B) and (A, C), they are recognized to be the same if the segments in A are aligned in time. In the constructed graph, nodes are time intervals in the particular utterance and edges are the distortion measures between the corresponding segments. An example of a clustering result is illustrated in figure 2.5. The clusters found in the graph are the discovered patterns. More on graph clustering can be found in the survey made by Schaeffer [Sch07]. Park and Glass used greedy bottom up clustering proposed by Newman [New04] in their work.

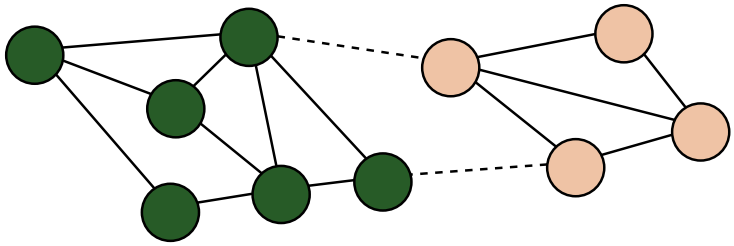


Figure 2.5: *An example of adjacency graph and clustering in it. The nodes represent exemplars and edges denote the similarity between those exemplars. The nodes are colored according to group membership.*

Finding patterns from speech and segmenting it in an unsupervised manner is possible without storing every exact actual exemplar or internal representation of discovered patterns. Discovery and segmentation can be achieved by merely building models of patterns relying on special statistics of atomic acoustic events. By using such statistics, the self-learning concept matrices (SLCM) algorithm is capable of acquiring representations of recurring word-like units from speech without any a priori knowledge. The ability to segment novel utterances in to word-like units is an inherent property of the algorithm. [Rä10]

The SLCM algorithm is based on the concept matrix (CM) algorithm [RLA09], which is slightly modified and appended with novelty detection to guide the process to create a new model or to update an existing one. The actual CM algorithm is described in more detail in section 2.4.1.

Oates has introduced the PERUSE algorithm [Oat02], which can discover frequently occurring patterns from multivariate time-series. The mean and variance of exemplars of patterns are used in model creation and recognition. PERUSE iteratively finds the number of frequently occurring unique patterns, their positions and lengths. It works in batch mode. All training material is expected to be available to it at all times. In addition to the main unsupervised learning task being iterative, the supervised parameter estima-

tion subproblem is using the expectation-maximization (EM) algorithm. These two properties can be considered as downsides. It is not reasonable to store large amounts of data and the algorithm becomes slow as the amount of data increases.

2.3 Supervised learning

Because a speech signal can be approximated as a piecewise stationary process, it can be modelled as a Markov process. A Markov process is a process that satisfies the Markov assumption: the next state of a system depends on only the current state, not the history of states. The state z_{n+1} is conditionally independent on states $z_1, \ldots, z_{n-1}$ given the state z_n. In some pattern recognition tasks like ASR the actual states are not observed directly but deducted from observations that are depended on the states. These kind of formalizations are called hidden Markov models (HMMs) [RS07]. In figure 2.6 a HMM is presented as a graphical model, which corresponds to the factorization presented in equation 2.11a. In the figure $z_1, \ldots, z_n$ are the hidden or the latent variables. In a m-state system each of these variables represent one of the m states: $z_1, \ldots, z_n \in \{1, \ldots, m\}$. The state transitions are defined by the $m \times m$ sized transition probability matrix $\mathbf{T}$, where the individual transition probabilities $T(i, j) = P(z_{k+1} = j | z_k = i)$ satisfy $0 \leq T(i, j) \leq 1$ and $\sum_{j=1}^{m} T(i, j) = 1$. In ASR systems generally $T(i, j)$ are large (close to 1) when $i = j$ and small (close to 0) when $i \neq j$, i.e. staying in the same state is more probable than changing the state. Each state i is characterized by the related emission probability ϵ_i, which captures the statistical properties of feature vectors (x) within the state. It is usually a Gaussian mixture distribution.

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_n, z_1, \ldots, z_n) = P(z_1)P(\mathbf{x}_1|z_1) \prod_{k=2}^{n} P(z_k|z_{k-1})P(\mathbf{x}_k|z_k) \quad (2.11a)$$

$$= \pi(z_1)\epsilon_{z_1}(\mathbf{x}_1) \prod_{k=2}^{n} T(z_k, z_{k-1})\epsilon_{z_k}(\mathbf{x}_k) \quad (2.11b)$$

A HMM periodically updates its state according to the state governing transition probabilities and on every update it emits an output, a feature vector, which is then observed. In ASR systems, the states are usually subword or sub-phoneme units. The feature vectors are some kind of spectral representation, e.g. MFCCs, of the frame. In the recognition the task is to find the most probable sequence of states (phonemes) that generated the

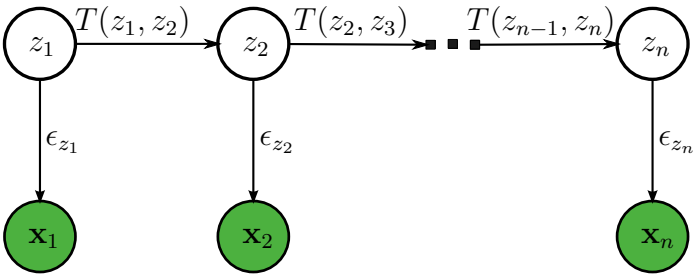


Figure 2.6: *A hidden Markov model*

sequence of feature vectors. Before the recognition is possible the HMM has to be trained. The training means estimating the three sets of parameters: the state transition probabilities $T(i, j)$, the emission probabilities ϵ_i and the initial state distribution π_1. Now the HMM in figure 2.6 can be formalized as the factorization in equation 2.11b.

A HMM for ASR can be trained in a fully supervised manner, but it is often very laborious since it basically means annotation and segmenting the speech samples by hand. After this data preparation, the transition probability and the initial distribution estimates are calculated from the annotation. The estimates of emission probability distributions are collected from speech segments corresponding to each state. In the case of a mixture of Gaussian distributions it means computing the mixture weights, mean vectors and covariance matrices for each mixture. Parameters for HMMs can also be trained also iteratively using the EM algorithm or modifications of it, for example [SW96].

With a very large training corpus the segmentation by hand is impossible due to time or financial constraints. There are algorithms that can optimize the alignment between the states and the feature vectors. The two most popular are the forward-backward algorithm [BPSW70] and the Viterbi algorithm [For73].

To be noted, a HMM can be trained also in an unsupervised manner (see e.g. [hSGC+14] and accompanying references).

2.4 Weakly supervised pattern discovery

All machine learning algorithms cannot be categorized strictly as supervised or unsupervised. Various terms like weakly supervised, semi-automatic, and self-supervised learning have been used to describe the methods falling outside the strict categorization. The usage of those terms is loose and they are used as both synonyms and to differentiate learning algorithms. In grammar

induction, for example, minor hand-editing between some iterations in an otherwise fully supervised learning algorithm could improve the quality of the grammar [WMS01].

The term weakly supervised learning is used to denote learning situations where training is not exact due to noisiness of labeling or the training exemplars. The training exemplars could contain extra information not related to jointly given labeling. Methods that can be considered as weakly supervised pattern discovery include but are not limited to non-negative matrix factorization (NMF) [LS01], acoustic dynamic programming ngrams (DP-ngrams) [Aim09], and the concept matrix(CM) [RLA09] algorithms. In the following subsections, NMF and DP-ngram's are briefly described whereas the CM algorithm is explained more closely because it is at the core of the methodology used in this thesis.

2.4.1 The concept matrix algorithm

The CM algorithm learns to recognize patterns from quantized time series without exact training in the traditional sense of supervised learning. In training it takes discrete symbolic data sequences and related context labels. It is called weakly supervised because the input sequences are known only to contain segment(s) that are instances of jointly given label(s). The lengths and positions of the instances are not known a priori. Through multiple training exemplars the excess portions of input sequences, not related to the label, average each other out and the future instances of patterns are recognized through their statistical properties.

The CM algorithm learns associations between discrete sequential data and categorical contexts related to the data. Once trained, the CM algorithm gives instantaneous probabilities of existence, or *activation levels*, of learned contexts for future untrained data. These training and recognition phases are defined in more detail in this subsection.

2.4.1.1 Training

The CM algorithm takes as input a set of discrete sequences $\mathbf{s}_i = [s_{i,1}, s_{i,2}, \ldots, s_{i,N_i}]$ and subsets of context labels $\mathbf{c}_i \subseteq C$, $\mathbf{c}_i = \{c_{i,1}, c_{i,2}, \ldots, c_{i,M_i}\}$ related to the sequences. N is the length of an discrete sequence and M is the size of a set. Typically $N \gg M$ for sequence-label-pairs.

To be noted that in basic CM algorithm $\mathbf{c}_i$ is a set. In a set the order of elements does not matter. Two set constituting of the same distinct elements are essentially the same. Additionally, sets cannot have duplicates.

From the sequences the algorithm collects frequencies of occurrences of element pairs with lags $\mathbf{k} = \{k_1, k_2, ..., k_K\}$ and stores them in frequency matrices $\mathbf{F}_{k,c}$. The training thus produces KN_C matrices of size N_sxN_s, one for each label $c \in C$ with every lag k. K is the number of different used lags, N_C is the number of all labels, and N_S size of the alphabet of the discrete sequences S.

The indexing of the symbols in the used alphabet can be used in the place of actual symbols. For example in the English alphabet:

$$\begin{aligned} s_1 &= a = 1 \\ s_2 &= b = 2 \\ &\vdots \\ s_{26} &= z = 26 \end{aligned}$$

This offers a convenient way of using them directly as indexes of matrices. Therefore, an element in a frequency matrix $\mathbf{F}_{k,c}(s_i, s_j)$ is the frequency of occurrences of element pair s_i, s_j separated by $k-1$ elements in all sequences related to label c.

Next step in training is the normalization of the frequency matrices to activation matrices $\mathbf{Q}_{k,c}$. First, transition probabilities from each element to every element with every lag are computed:

$$\mathbf{P}^S_{k,c_n}(s_j|s_i) = \frac{\mathbf{F}_{k,c_n}(s_i, s_j)}{\sum_{j=1}^{N_S} \mathbf{F}_{k,c_n}(s_i, s_j)} \tag{2.12}$$

which is the probability of element s_j of being k:th element after element s_i in all label c_n related sequences when intermediate elements are not considered.

The second step in normalization is the computation of conditional transition probabilities $P(s_i, s_j \in c_n \mid k)$ to activation matrix:

$$\mathbf{P}^C_{k,c_n}(s_j|s_i) = \frac{\mathbf{P}^S_{k,c_n}(s_j|s_i)}{\sum_{m=1}^{N_C} \mathbf{P}^S_{k,c_m}(s_j|s_i)}, \tag{2.13}$$

which is the probability of transition from s_i to s_j occurring with lag k in the presence of label c_n instead of the same transition occurring in the presence of any other label.

Equation 2.13 already gives the maximum likelihood estimate for context label c given transition from s_i to s_j. Normalization is taken one step further and is finalized by subtracting $\frac{1}{N_C}$ from conditional probabilities to force the sum of activations of all possible labels c to zero at all times:

$$\mathbf{Q}_{k,c} = \mathbf{P}^C_{k,c} - \frac{1}{N_C} \tag{2.14}$$

The reasoning behind this idea is that if the sequences S are generated by an independent and identically distributed random process, all the activation levels would equal this value. Therefore all activation levels below $\frac{1}{N_C}$ are considered meaningless.

2.4.1.2 Recognition

For an input sequence the CM algorithm gives instantaneous activation levels for every trained context. The element pairs with different lags in input sequences are used as pointers to previously created activation matrices $\mathbf{Q}_{k,c}$. The activation level of a context c_n at time t is the sum of the normalized activation values at all learned lags:

$$A(c_n, t) = \frac{1}{K} \sum_{m=1}^{K} \mathbf{Q}_{k_m}(s_t | s_{t-k_m}, c_n) \tag{2.15}$$

The context having highest activation level at a certain time instance is considered the context that most likely generated the present input. Usually it is convenient to examine temporally longer windows if contexts are known to be distributed in time.

2.4.2 Non-negative matrix factorization

NMF algorithm can also be used in weakly supervised learning framework for training of word models for speech recognition [VH08]. In the NMF, a matrix $\mathbf{V}$ of size $n \times t$ is approximated as a product of matrices $\mathbf{W}$ and $\mathbf{H}$ of sizes $n \times r$ and $r \times t$ respectively:

$$\mathbf{V} \approx \mathbf{WH}, \tag{2.16}$$

where $\mathbf{V}$ is multivariate n-dimensional data with t observations. As the equation suggests the i:th column of $\mathbf{V}$ is the linear combination of column, or *basis*, vectors in $\mathbf{W}$ with multiplying coefficients, or *model activations* in corresponding i:th column vector in $\mathbf{H}$.

The name for the NMF comes from the constraint that all elements of all matrices in the factorization have to be strictly zero or positive. $\mathbf{V}$ Some computationally effective methods to solve matrices $\mathbf{W}$ and $\mathbf{H}$ in equation 2.16 are presented in [LS01].

Usually r is chosen so that $(n+t)r < nt$, which enforces the reconstructed matrix $\mathbf{V}$ to lower dimensionality. When $r << t$ a high quality approximation is achieved only when latent structure is discovered from the data. In Van Hamme's experiments the latent structure is the limited vocabulary of

a large corpus of t utterances, namely the TIDIGIT [Leo84] corpus [VH08] [Van08]. For the use of NMF utterances have to be presented with constant dimension feature vectors. These vectors are the t column vectors of $\mathbf{V}$ in equation 2.16. r is set to slightly greater value than the number of words in the training corpus to allow training or discovery of silence or noise.

2.4.2.1 Training

A NMF in its basic form as presented in equation 2.16 operates in unsupervised manner [SDVh08]. Weak supervision can be included to training by extending equation 2.16 by information matrix $\mathbf{G}$ [VH08]:

$$\begin{bmatrix} \mathbf{G} \\ \mathbf{V} \end{bmatrix} = \begin{bmatrix} \mathbf{W}_g \\ \mathbf{W}_v \end{bmatrix} \mathbf{H} \tag{2.17}$$

$\mathbf{V}$ is a collection of non-negative fixed-length representations of utterances, which can be constructed for example from phone lattice transition probabilities. $\mathbf{G}$ is a $m \times t$ matrix, where an element with index i, j denotes how many times i:th word is contained in the j:th utterance. This weak supervision is conceptually similar to the labeling done in the training of basic CM algorithm. The actual training equals to the computation of factorization in equation 2.17. The outcome is collection of model vectors as latent variables for acoustic and information data: $\mathbf{W}_v$ and $\mathbf{W}_g$.

2.4.2.2 Recognition

The recognition on unseen data is performed first by computing $\hat{\mathbf{H}}$ from $\mathbf{V} \approx \mathbf{W}_g\hat{\mathbf{H}}$ with $\mathbf{W}_v$ obtained from training. Second, the word activations in this unseen data are estimated with following equation:

$$\hat{\mathbf{G}} = \mathbf{W}_g\hat{\mathbf{H}}, \tag{2.18}$$

where column vector of $\hat{\mathbf{G}}$ expresses to which extent each learned word is found in corresponding test utterance.

2.4.3 Acoustic DP-ngram

Aimetti has proposed a weakly supervised method for learning keywords from utterances [Aim09]. It is a modification from DP-ngram method originally developed to find similar segments of DNA sequences [SK83] and hence called *acoustic* DP-ngram.

The segmentation of utterances to extract keyword instances is based on finding optimal alignments between pairs of utterances and is similar

to the DTW described in the subsection 2.2.3. Instead of comparing every utterance to every other utterance a soft labeling is used to force the discovery of the local alignments. The optimal local alignments are obtained only from the utterance pairs sharing the same keyword label. This allows the method either to build full list of obtained word exemplars or to calculate 'centroid' exemplars/prototype models for every keyword. The recognition, or the keyword detection from utterances, is reported to perform better with the full lists of exemplars than with the 'centroids'. In the experiments it seems that the 'centroids' based methods cannot handle the variability of acoustic speech signal and the word detection capability starts to decline when 'too many' utterances are observed.

2.5 Multimodal pattern recognition

The definition and the use of the word multimodal varies along different researchers and studies. The multimodality of data brings both challenges and opportunities. The expectation is the possible leverage to machine learning tasks that could be obtained from extra information provided by extra modalities. In many cases it is not feasible or even possible to fuse data together to multivariate data and use traditional techniques. This could be due to a few reasons: the nature of the data streams are so different that same recognition/discovery method cannot be used or the streams could be temporally and/or spatially asynchronous. For example in industry fault diagnostics there could be strong association between two sensors of different kinds at different parts of the process, but with variable delay.

2.5.1 Statistical machine translation

A statistical machine translation (SMT) is a machine learning algorithm that learns to translate from one natural language to another. Usually a SMT algorithm is trained by giving samples of human-produced translations—parallel corpus—for example official documents from the European Union [Koe05]. The SMT algorithm seeks for patterns from these large amounts texts. Based on the learned patterns and their statistics it makes intelligent guess what the appropriate translation could be. [Lop08]

The two main problems that a SMT algorithm has to solve are the ambiguity of word translation and the differing word ordering between the source and the target languages. Some words can be translated without loss of meaning to different morphological variants of the same word, while some other words have translations with totally different meanings. The source

and the target language could have typically different syntactic structure, so some sort of reordering has to be applied. [Lop08]

The development of a SMT system generally contains the following steps:

1. construction of a translational equivalence model
2. parametrization
3. parameter estimation
4. decoding

Formally the task of a machine translation system is to transform a sequence of words from a source language to a sequence of words in the target language. The goal is to find a translationally equivalent I words long sequence $e_1^I \in {V_E}^I$ for a given J words long input sequence $f_1^J \in {V_F}^J$. The translational equivalence models are not capable of providing single unambiguous sequences. A parametrization is assigned to the model in order to resolve the ambiguity, allowing the rating of source and target sequence pairs. The parameters in parametrization are estimated by means of machine learning methods. The final step is the actual translation, also called decoding, which equals the search for the highest rating target sequence according to the model. [Lop08] More on the parameter estimation can be found in [BPPM93].

Translational equivalence models are important since they reduce the search space in which translations are searched for in the decoding phase. One of the two most used translational equivalence models is synchronous context free-grammar (SCFG), which is a generalization of a CFG [Lop08]. Instead of producing one output string it produces two output strings and defines an alignment between them. Co-indexed nonterminals are used to define correspondence between strings. When two nonterminals share the same co-index, they are aligned. Below is a section of a SCFG. Each production produces two outputs, one for each language. Here they are separated by a slash (/).

Table 2.1: *A section of a SCFG for English and Japanese*

S_{10}	$\rightarrow$	$NP_{11}VP_{12}$ / $NP_{11}VP_{12}$
VP_{12}	$\rightarrow$	$V_{13}NP_{14}$ / $V_{13}NP_{14}$
NP_{11}	$\rightarrow$	i / watashi wa
NP_{14}	$\rightarrow$	the box / hako wo
V_{13}	$\rightarrow$	open / akemasu

Applying production rules to any aligned nonterminals until there is not any nonterminals left produces English and Japanese strings which are translations of each other (i open the box / watashi wa hako wo akemasu). [Chi06]

A way to approach SMT is generative and probability models through the chain and the Bayes' rules:

$$P(\mathbf{e}|\mathbf{f}) = \frac{P(\mathbf{e},\mathbf{f})}{P(\mathbf{f})} = \frac{P(\mathbf{f}|\mathbf{e})P(\mathbf{e})}{P(\mathbf{f})} \tag{2.19}$$

In SMT $P(\mathbf{e})$ is called the language model and $P(\mathbf{f}|\mathbf{e})$ the translation model. Applying the translation equivalence model to this decomposition reduces the search space. Additionally, the maximization problem of the conditional probability $P(\mathbf{e}|\mathbf{f})$ is independent of language model $P(\mathbf{f})$. With these two modifications the following decomposition is obtained:

$$P(\mathbf{e},\mathbf{d}|\mathbf{f}) \propto P(\mathbf{f},\mathbf{d}|\mathbf{e})P(\mathbf{e}) \tag{2.20}$$

Further assumptions have to be made to make the search for the most probable estimate $\hat{e}^I$ more tractable. These include, for example, assuming that a word in either language is conditionally dependent only a certain amount of words before it. Moreover, the approaches are not limited to generative models. More details on SMT can be found for example in [Lop08].

Loosely speaking the source and target languages can be considered two modalities of the same higher level meaning, which is considered when SMT is resolving the translational equivalence. The need for reordering is the main feature that differentiates SMT algorithms from the method later presented in this thesis.

Two popular examples of freely available web services based on SMT are Google Translate [Goo12] and Microsoft Translator [Mic12].

2.5.2 Other multimodal learning paradigms

Probably the most obvious example of a multimodal data stream is multimedia, which can be a combination of video, audio, text, and still images. Much of the multimodal research is done around segmentation of such data, which is also called scene boundary detection. These segmentation methods get leverage from the existence of multiple modalities. For example Xie proposes methods in her dissertation that are able to find patterns that reside across multiple modalities. Her methods were aimed at detecting dense structures from multimedia. In the experiments the methods were able to discover different semantic segments, like "play" and "break" of soccer and baseball

games using both visual and acoustic cues. Furthermore, news broadcasts were segmented with additional textual modality. [Xie05]

In Xie's approach, the recurring events in video are modelled as HMMs, and the higher level transitions between the events as another level of a Markov chain. This structure creates a hierarchical hidden Markov model (HHMM). In the experiments the higher level HMM corresponds to semantic events, like 'play' and 'break' in a soccer game, and the lower level corresponds to the occurring variations within the same event. Together with automatic feature and complexity selection this unsupervised method achieved comparable or even better results when compared to both supervised and unsupervised HMMs. [Xie05]

Ballard and Yu proposed a method [BY03] for word acquisition that makes use of raw multimodal data. In their experiments users explained actions they were performing, for example *"Stapling the paper"*. Through phoneme recognition the utterances are transcribed to phoneme strings. In conjunction with spoken utterances, data was collected from multiple user mounted sensors. The sensors captured body movements and gaze fixations. From the additional sensor data two contextual data streams were constructed: actions and attentional objects. Different actions and objects are used as soft labeling for co-occurring phoneme strings. From any pair of phoneme strings that share the same label, similar sub-strings are extracted and clustered together. Under the same label multiple clusters are formed, because a phoneme string can contain multiple labels (e.g. an object and the action on the object). Each cluster represents a hypothesized lexical item for the label. Finally, the expectation-maximization (EM) algorithm is used to find the most probable lexical item given the label. [BY03] The method proposed by Ballard and Yu is an unsupervised word learning and grounding method. It does not provide predictions of the outcome of one modality from another or in other words transcription ability.

Ridge et. al. proposed an unsupervised learning method that builds an internal representation between basic objects and their affordance [RSL10]. The term affordance characterises the action possibilities that an environment offers an agent acting in the environment [Gib86]. The algorithm is evaluated in experiments with a robotic arm that interacts with household objects on a table. The two modalities used in the experiments are still images taken prior to the interaction and video after a pushing action of the arm. The input features describe the image of the object and the output features describe the movement of the object. The proposed algorithm learns to classify novel objects in to affordance classes. The approach is somewhat similar to the method proposed in this thesis: the found structure in one modality together with the co-occurrence information, are used in the labeling of supervised

construction of a classifier in the other modality. Ridge et. al. use a cross-modal neural network that has two layers of codebook vectors fully connected via a Hebbian weight mapping. The layers are trained trough Kohonen's self-organizing maps and a variant of learning vector quantization algorithms [Koh97]. The co-occurrence is exploited in the training of the Hebbian map. The Hebbian links between the best matching unit nodes in the two layers are updated every time a new feature vector pair is introduced to the system.

Chapter 3

Cross-modal statistical learning

In this chapter the methodology for automatic self-supervised learning of associations is proposed. The process is described in the light of the given task of maximizing the predictability of the textual representation when the learner is given corresponding speech audio. Part of this work can also be seen in a paper published in the conference proceedings of the Interspeech 2013 [KRL13].

The individual algorithms for each given sub-task in the learner are selected using domain expertise. The selected algorithms are proven to be effective in their domains. The main novelty aspect here are how they are combined and that information on the ordering and the length of patterns in textual modality is used to segment feature representation of speech. The learner acquires the transcription ability only from speech to text, because some of the key algorithms are not generative.

First, an overview of the whole algorithm is given and continued by defining the research problem in precise mathematical form. Throughout the remainder of this thesis, the same notation is used at different stages of the whole algorithm

3.1 The overview

As mentioned in chapter 1, patterns are first discovered in one modality and then used as labels in the training in the other modality. The selection of the CM algorithm for pattern recognition in speech audio guides further algorithm selections. The CM algorithm needs labels in the training phase and the textual representations of the speech audio is a good source for labeling, since it is noiseless in the sense that instances of the same patterns are always the same, assuming the used corpus is free from typographical

errors.

The CM is selected for pattern recognition from audio, because it has proven to be effective in real keyword discovery when experimenting with the same CAREGIVER [AtBA+10] corpus used in this thesis [RL12]. The novelty value of this thesis is that keywords are not known a priori and utterances are evaluated as a whole. The CAREGIVER corpus is presented later in this chapter.

Since the textual modality is basically strings of concatenated characters, the actual words that constitute each utterance are not know a priori. The approach is to apply statistical inference to the collection of utterances presented as continuous text to create an applicable grammar. The created grammar is used to extract patterns, or word-like units of text, from each utterance. These patterns are then used as labels in the CM algorithm. For building the grammar a variant of HML is used.

With the CM algorithm, labeled patterns can be recognized from speech audio. These ordered sets of labels are transcribed back to their textual representations and then concatenated. This text string is a *hypothesis*, which is evaluated by comparing it to the ground truth, the corresponding *reference* text string. The fidelity of the hypothesis is evaluated with edit distance. Generally edit distance refers to a string metric called the Levenshtein distance [Lev65], which measures the difference between two symbolic sequences.

3.2 The research material - CAREGIVER corpus

The signal pairs used in the experiments are derived from CAREGIVER corpus [AtBA+10]. It is composed of infant directed speech as high quality speech audio utterances accompanied by their orthographical transcripts. The whole corpus is multilingual, but only the English portion of it is used. The original purpose of the corpus is to study keyword learning from speech. It contains utterances with 1-4 keywords placed in a carrier sentence. The keywords are statistically balanced, which produces grammatically correct sentences that are not necessarily semantically plausible. Since there should not be strong word-to-word dependencies between the keywords, the fact could be used in the evaluation of the stopping criterion in the grammar induction for continuous text. The iteration should self terminate before the algorithm starts form such production rules that lead to text strings containing two keywords.

In the recordings of the utterances 10 speakers were used: four main

speakers and six additional speakers. Each of the four main speakers have 2397 utterances and the additional speakers 600 utterances each. The corpus also contains single productions of keywords. In these experiments one main talker was used.

3.3 Problem formulation

The discovery of the associations between two sequential data streams that are hypothesized to contain mutual dependencies is studied. These dependencies are very weak at the level of raw data due to their nature. Instead, the strong dependencies are assumed to exist only between higher-level patterns extracted from both data streams. The data is represented as a set of pairs of signals $\{\mathbf{s}_T, \mathbf{w}_T\}$ from training sets S and W, denoting speech audio and string of characters respectively. The sub-index T denotes the number of utterance $T \in [1, 2, \ldots, N]$, where N is the total number of utterances.

After being given these N pairs for training, the learning agent is expected to acquire the ability to produce estimates for $\mathbf{w}$ when given a novel spoken utterance $\mathbf{s}$.

3.4 Pre-processing of the speech audio

For the use of the CM algorithm the speech signal is transformed into a discrete signal that represents the atomic acoustic events. The discrete signal is the result from computation and clustering of MFCCs [DM80]. In the experiments for this thesis 12 MFCCs from 32-ms long windows are extracted at 10-ms intervals. These 12-dimensional continuous value vectors are clustered to $N_A = 128$ clusters with k-means and the vectors are replaced by their corresponding cluster IDs. This gives us the vector quantized (VQ) symbolic speech sequence with alphabet size of N_A at 100 Hz symbolrate (see chapter 2). A similar pre-processor consisting of MFCC vectors and clustering is used in many other speech processing related studies [Rä11] [Van08].

3.5 Pattern discovery from continuous text using greedy agglomeration

To discover patterns and use them for labeling in the CM algorithm, grammatical inference is used. The textual representations of training utterances are concatenated, and all the special characters—including white spaces—are

removed. This long continuous text string is then compressed by agglomeration and a CFG built as explained in section 2.2.2. The set of labels are the word-like sequences of text corresponding to the non-terminal symbols in the compressed sequence. The labels and the discovered patterns to be associated with each individual VQ speech utterance are the ones recognized by the induced grammar. Each utterance is compressed using the agglomeration rules in the same order they were selected in the induction phase. When there are no more possible agglomeration rules to be used, the patterns are the word-like sequences corresponding the non-terminal symbols.

3.6 Segmenting VQ speech sequences

In this study the contextual labels $\mathbf{c}$ are the text patterns that are recognized from the continuous text with the previously acquired CFG. As stated in [RL12], these contexts could be used as either an unordered or ordered set in the input for the CM algorithm. Since the two streams (speech and text) are different representations of the same phenomenon, a priori knowledge is exploited in that the patterns are in the same order in both representations. VQ representations can be segmented in order to discard the VQ elements that are almost surely not related to the text pattern.

A reasonable constraining assumption is that the pattern boundaries are approximately in the same relative positions in the both representations. Due to the nature of the English language and the existence of silence periods in speech, it cannot be guaranteed that all the relevant VQ elements are between the boundaries suggested by the corresponding text pattern (the dashed vertical lines in figure 3.2). For this reason the length of VQ speech sequences used in the training, together with a text pattern is extended by an experimental factor of 1.4 . For example if a VQ speech segment suggested by the text pattern is $v_i v_{i+1} \dots v_{i+n-1}$, where v is a VQ element and n is the length of the segment, then the extended segment is $v_{i-0.7n} \dots v_{i+1.7n-1}$ (the highlighted portion of the sliced bar representing VQ speech sequence in figure 3.2).

For example, if a pattern c_i is found in signal $\mathbf{w}_T$, its relative length in that utterance is $\frac{|c_i|}{|\mathbf{w}_T|}$. Since the two signals, $\mathbf{w}_T$ and $\mathbf{s}_T$, are more or less asynchronous, relatively longer segments of $\mathbf{s}_T$ are taken to be associated to context tag c_i. An extension factor of 1 is used to both ends of the pattern in the results presented here. So, relative length of the associated segment of $\mathbf{s}_T$ is up to three times as long as the relative length of c_i.

3.7 the concept matrix (CM) algorithm

After acquiring the VQ segment-label pairs, they are used to train CMs as described in section 2.4.1.

3.7.1 Recognition

The recognition is done in two phases. The idea is to look for patterns that have high activation on short and long temporal intervals. In other words, two requirements have to be met for a pattern to be included in to the hypothesis. First, it has to have high integral of activation values across the whole sequence, and second, it also has to be the most probable candidate across some much shorter temporal interval. The formation of a hypothesis goes as follows.

First, a short-term sliding window is used to accumulate CM activation value outputs $A(c,t)$ into local pattern probability estimates. The integral of activation values for pattern c at time t with window length T is:

$$A_{tot}(c,t) = \sum_{t-T/2}^{t+T/2} A(c,t) \tag{3.1}$$

The pattern that has the highest integral across the window is selected for the hypothesis for this short time interval. Application of the sliding window to activation values $A(c,t)$ with step size T_S results in an ordered sequence of pattern hypotheses $\tilde{\mathbf{c}}$ of length $\frac{|A(c,t)|-T}{T_S} + 1$. Second, an unordered set of pattern hypotheses $\tilde{\mathbf{c}}_\tau$ is formed by measuring the overall activation of all patterns across the entire utterance and then retaining the patterns that exceed a pre-defined activation threshold τ.

By taking an intersection $\tilde{\mathbf{c}} \cap \tilde{\mathbf{c}_\tau}$, preserving the order of the hypothesized patterns, and merging consecutive duplicates of detected patterns, the final hypothesis of most likely textual representation $\tilde{\mathbf{w}}_T$ is obtained by converting the hypothesized text patterns into their textual representations according to the production rules in the induced CFG.

3.8 Evaluation the quality of the hypothesized sequences

In the evaluation of the fidelity of a single hypothesized utterance, a variant of common Levenshtein distance [Lev65] is used. The Levenshtein distance

measures dissimilarity of two strings in the form of the minimum amount of single character edits (deletions, insertions and substitutions) needed to convert a string to another string, here corresponding to the hypothesized utterance and the corresponding reference. Here, the variant is referred as *edit distance* and in it deletion and insertion have equal weights (1) and substitution twice that weight (2). For example, edit distance between *bargain* and *jargon* is 5, resulting from two substitutions and one deletion (See table 3.1 for more detailed computation). Furthermore, the edit distance is zero for a hypothesis matching the reference exactly. The goodness of one individual edit distance measure depends naturally on the length of the reference. In order to compare edit distances of utterances of different lengths, individual edit distances are normalized with the corresponding reference utterance length. The resulting measure is referred as *relative edit distance (RED)*.

Edit distance between two strings a, b can be computed using Wagner-Fischer algorithm [WF74]. It is computed recursively using equation 3.2 and given by $\mathrm{lev}_{a,b}(|a|, |b|)$, where

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} \max(i,j)) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \mathrm{lev}_{a,b}(i-1,j) + 1 \\ \mathrm{lev}_{a,b}(i,j-1) + 1 \\ \mathrm{lev}_{a,b}(i-1,j-1) + 2 * [a_i \neq b_j] \end{cases} & \text{otherwise} \end{cases} \tag{3.2}$$

The first element in the min-part of equation 3.2 corresponds to deletion from a, the second insertion, and the last substitution in the case of character mismatch. The recursion visually corresponds filling a $(|a| + 1) \times (|b| + 1)$ sized matrix element-by-element. The computation of edit distance for words *bargain* and *jargon* is illustrated in Table 3.1.

In the experiments, the optimization task is to maximize the predictability of the textual representations for a set of spoken utterances. That corresponds to minimizing the sum of individual edit distances of the hypotheses in the set, which is referred to as the sum of edit distances (SED).

Table 3.1: *Edit distance computation example using equation 3.2 recursively. The performed single character operations are listed on the right side of the matrix and corresponding cumulated edit distance values are emphasized and underlined in the matrix.*

		j	**a**	**r**	**g**	**o**	**n**		
	0	1	2	3	4	5	6		
b	1	*2*	3	4	5	6	7	substitute	**b** → **j**
a	2	3	*2*	3	4	5	6	match	**a** = **a**
r	3	4	3	*2*	3	4	5	match	**r** = **r**
g	4	5	4	3	*2*	3	4	match	**g** = **g**
a	5	6	5	4	3	*4*	5	substitute	**a** → **o**
i	6	7	6	5	4	*5*	6	delete	**i**
n	7	8	7	6	5	6	*5*	match	**n** = **n**

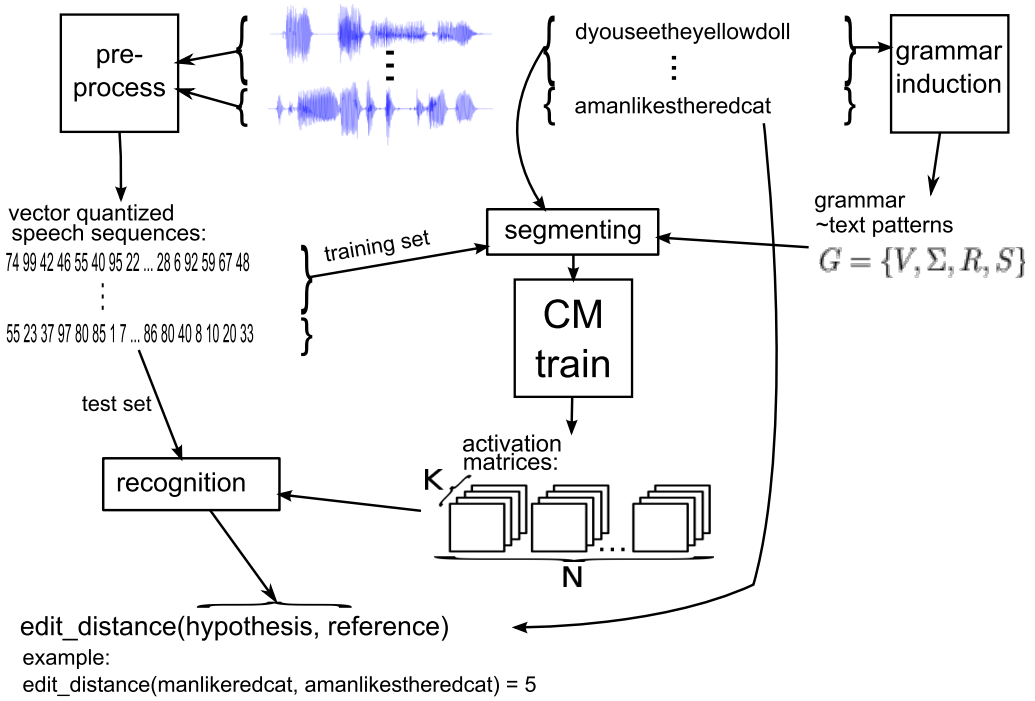


Figure 3.1: *Overview of the whole process of pre-processing, training, and recognition. The process starts with pre-processing utterances from high quality wav-audio signals to vector quatized sequences and inducing grammar from corresponding utterences in textual representation. The training set of vector quantized sequences are segmented using the induced grammar. The CMs are trained with terminals (labels) and corresponding segments. Using the CMs, the hypotheses for textual representations of VQ speech test sequences are formed by recognizing labels inside each VQ speech sequence. Finally, calculating the edit distance between hypothesis and corresponding reference gives the quality of recognition*

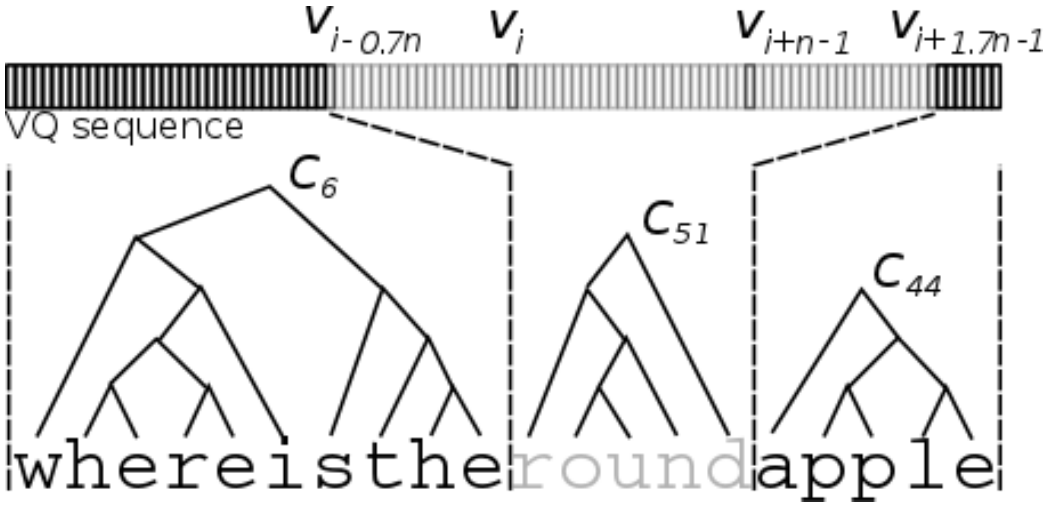


Figure 3.2: *The text patterns are recognized from the root nodes (the C's) of CFG bottom up parse trees. The CM is trained with longer segments of VQ representation than text the pattern suggests. The segment is extended 70 percents from both ends.*

Chapter 4

Experiments

The proposed methodological framework was developed and tested in the MATLAB environment. In the conducted experiments, the learning algorithm is given a training set of utterances with the textual and VQ speech representations in order to build the statistical association across the two representations. The measure of the success of pattern discovery and associative learning is the fidelity of the textual representations that are derived from the given VQ representation of speech from a disjoint test set.

4.1 The material

The material used in the experiments is taken from the CAREGIVER corpus [AtBA$^+$10]. The corpus contains infant directed spoken utterances in multiple languages and speakers. Along the high quality speech signals there are corresponding orthographic transcripts of each utterance. From the Y2-version of the corpus 2397 utterances of a single English speaker are used in the experiments. Each of these utterances contains from one to multiple words. The utterances are constructed from a list of 50 keywords and accompanying carrier phrases. The keywords are statistically balanced which causes some of the utterances to be logically absurd but still every utterance is grammatically correct.

4.1.1 Data preparation

The utterances are rearranged in to a random order. After the rearrangement the first 2000 utterances are used for parameter evaluation and training. The remaining 397 utterances are used as the test set. For consistency, throughout the experiments both the order of the utterances, as well as the division to

training, evaluation and test sets are kept the same.

4.1.2 Extraction of label - VQ sequence pairs for the CM algorithm

The textual representations of utterances in the training set are concatenated, and the grammatical inference—described in the subsection 2.2.2—run until the stopping criterion in the equation (2.7) is met. The resulting CFG is used in bottom up parsing of individual utterances. The root nodes representing the word-like units of text are then used as labels in the CM algorithm. The information about the portions that the recognized patterns occupy in the textual representations is used in extracting segments from the corresponding VQ representations. The label recognition and segmentation of VQ representation is illustrated in figure 3.2. Since the two vectors in different modalities are known to contain the same information, it is reasonable to assume that the length of VQ speech segment is proportional to the length of the label. The lack of direct correlation is taken into account by pairing a 1.4 times longer segment of VQ speech than corresponding label length would suggest. This guarantees that the VQ speech segment (almost surely) contains the actual VQ speech representation of the label, but still some excess VQ speech is dropped out. The extraction is performed in all training utterances and with all the discovered labels. The CM matrices are trained with resulting label-VQ segment pairs using lags: $k \in \{1, \ldots, 13\}$.

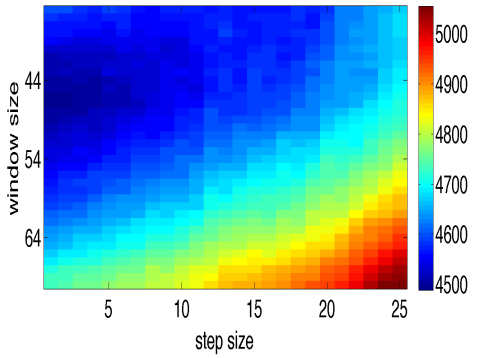
In testing, the hypothesized textual representations of utterances are derived from the activation graphs, which are the resulting output from the CM algorithm when given a VQ speech sequence. These representations are compared against corresponding reference strings to evaluate the fidelity of the hypotheses, i.e. predictability.

4.2 The results

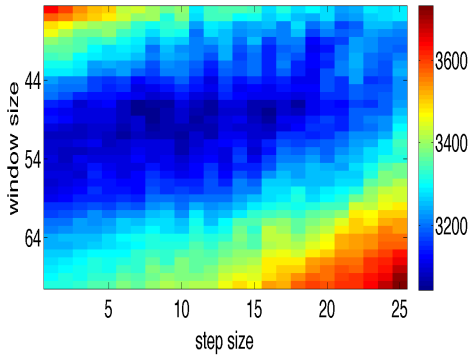
The experiments were run with three different CFG variants. The variants are the result of different selection criteria of symbol pairs in the inference of CFG. These variants are Freq, MI and IG as presented in section 2.2.2.1.

The success of the estimation with each variant is evaluated by comparing the estimated parameters to the optimal ones. The variants are compared to each other with the SED measure.

The parameter values for sliding window length T and for the step size T_s to be used in the final testing on the independent test set are estimated from the training set using a 5-fold evaluation procedure. In each fold, 1600



(a) *Heat map for parameter estimation. Estimates* $T = 47$ *and* $T_S = 1$ *samples correspond the minimum SED of the map.*



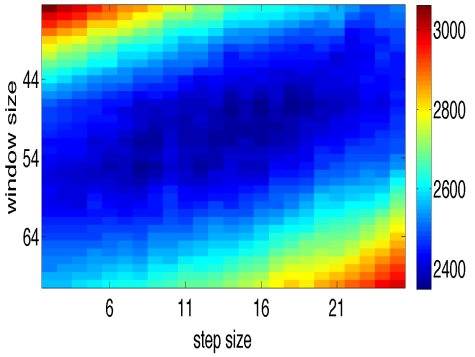
(b) *Heat map of SEDs in the test set. The SED with estimated parameter values is 3144. The optimized SED is 3044 and is given by* $T = 48$ *and* $T_S = 7$ *samples.*

Figure 4.1: *Freq heat maps. Each tile's color represents the SED with corresponding sliding window size* T *and step size* T_S*.*

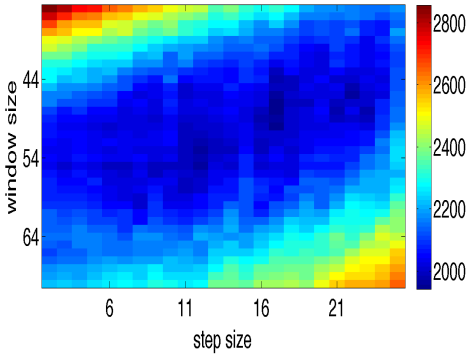
utterances are used for training and 400 utterances for evaluation of the text reconstruction performance. The parameter values are varied and for each combination of values the SED is computed and averaged over the folds. The parameters' value estimates are the ones that gives the minimum average SED within the evaluation sets. Similarly, for the test set of 397 utterances the SED measures are computed for the same parameter value combinations. These computation results for Freq, MI, and IG variants are visualized as heat maps in figures 4.1, 4.2, and 4.3 respectively. Since here the task is to minimize, colder (bluer) is better. In the figures (a)-tiles are the averaged heat maps from the 5-fold evaluation and (b)-tiles heat maps representing SEDs with different parameter values in the test set.

The results extracted from the heat maps are summarized in table 4.1. All the estimated parameter values and the corresponding SEDs are compared to the corresponding optimized values. In all variants the parameter values are quite reliably estimated since the SEDs given by those estimated values differ only 2.4%-5.1% from the optima. In terms of SED the IG variant outperforms the MI variant by 8.7% and the Freq variant by 40.6%.

The individual edit distances with the estimated parameter values of the best performing—the IG variant—are examined in more detail. 106 (26,7%) of the hypothesized utterances produced by the variant were correct. The whole test of 397 utterances is composed of 7786 characters. The SED being 1868 means that 76% from all the hypothesized characters were correct in

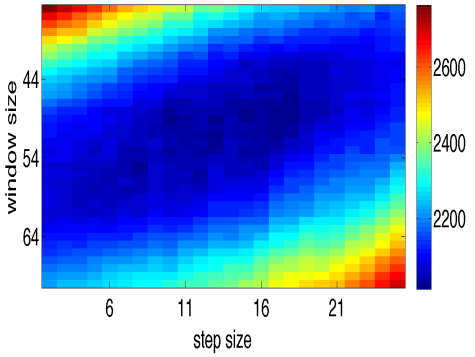


(a) *Heat map for parameter estimation. Estimates* $T = 48$ *and* $T_S = 14$ *samples correspond the minimum SED of the map.*

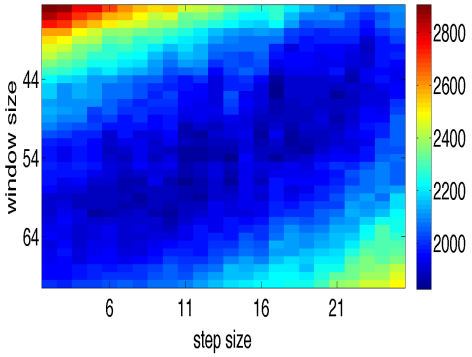


(b) *Heat map of SEDs in the test set. The SED with estimated parameter values is 2045. The optimized SED is 1942 and is given by* $T = 47$ *and* $T_S = 17$ *samples.*

Figure 4.2: *MI heat maps. Each tile's color represents the SED with corresponding step and sliding window sizes.*



(a) *Heat map for parameter estimation. Estimates* $T = 48$ *and* $T_S = 17$ *samples correspond the minimum SED of the map.*



(b) *Heat map of SEDs in the test set. The SED with estimated parameter values is 1868. The optimized SED is 1826 and is given by* $T = 46$ *and* $T_S = 17$ *samples.*

Figure 4.3: *IG heat maps. Each tile's color represents the SED with corresponding step and sliding window sizes.*

Table 4.1: *Summarized results with the three variants. The table includes the parameter value estimates from the 5-fold evaluation and the corresponding SED in the test set. For comparison the optimized parameter values for the test set and corresponding SED are also presented*

		window size	step size	SED of the test set
Freq	estimated	47	1	3144
Freq	optimized	48	7	3044
MI	estimated	48	14	2045
MI	optimized	47	17	1942
IG	estimated	48	17	1868
IG	optimized	46	17	1826

the test set. In utterances that are not hypothesized perfectly, on the average 62% of the characters were correct.

The effect of the length of the reference utterance to the edit distance is also studied. The Pearson's linear correlation coefficient for reference utterance length and edit distance in the whole test set is 0.2634, which can be regarded as low. In figure 4.4 the REDs are ordered in ascending order by the reference utterance length. From the figure can be seen the effect of the length of the utterance to the prediction accuracy. The variation in the recognition of single word utterances is large. Many of the hypotheses are correct and some are way off. In general the length of the multi-word utterances do not have a remarkable effect on the prediction accuracy. This is illustrated in the figure by moving median of 30 samples (bold dashed line) and the median over the whole test set (0.182, solid straight line).

Since the variants produce such different prediction accuracies, the inferred CFGs are compared in the table 4.2. The correlation between the grammar size, the coverage and the prediction accuracy can be seen. The used stopping criterion of the inference seems to work better for IG and MI variants than for Freq variants.

For a reliable comparison, the three variants are compared with the same coverage. The grammatical inferences in IG and Freq variants are continued until the resulting grammars reach the coverage of MI variant (≥ 0.9939) in the previous experiment.

The coverage of MI variant (≥ 0.9939) in the previous experiment is used as the stopping criterion of the inference in the IG and Freq variants. The results are presented in table 4.3. The order of the variants remains the same, but Freq variant improves significantly. Also IG variant improved slightly as the result of improved coverage. These results imply two things.

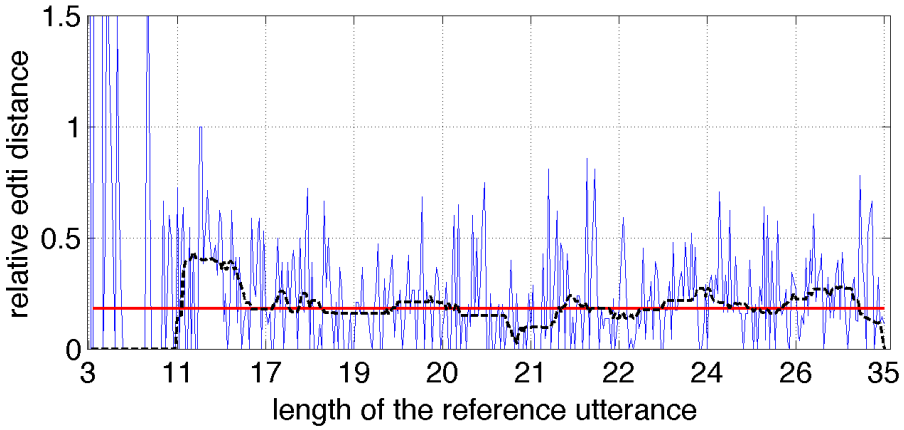


Figure 4.4: *Relative edit distances (REDs) ordered by the corresponding reference utterance lengths. The individual edit distances are plotted with a thin blue line. The bold dashed line is median filtered smoothing with a 30 sample long window. The straight line is the median of relative edit distances in the test set. Note that x-axis is not linear, the tick labels denote the length of the reference utterance in that index.*

First, the coverage is potentially a better stopping criterion than the one used in this study. The usage of the coverage as stopping criterion requires a priori knowledge on the quality of the data it is applied to. Second, the usage of IG in the place of MI in the iterative grammar induction results a grammar that is more suitable for recognition purposes, as was suggested in [WMS01].

Figure 4.5 illustrates the optimal SED as a function of grammar size. Generally it is safe to state the order of the variants is IG, MI, and Freq in terms of relative performance.

For the record, it should be stated that the MDL was briefly tested as

Table 4.2: *Comparison of CFGs inferred from the training set. Grammar size is the number of production rules. Patterns is the number of distinct non-terminals in fully compressed sequence. Coverage is the portion of original sequence that the non-terminals in the compressed sequence correspond.*

	Grammar size	Patterns	Coverage	weighted mean of pattern length
IG	190	103	0.992	5.17
MI	180	99	0.994	4.93
Freq	100	79	0.868	3.72

Table 4.3: *The comparison of the three variants' parameters and performance with their CFGs are reaching approximately the same coverage in the training set.*

	Coverage	Window size	Step size	Optimum SED
IG	0.9944	55	16	1777
MI	0.9939	47	17	1942
Freq	0.9942	56	16	2033

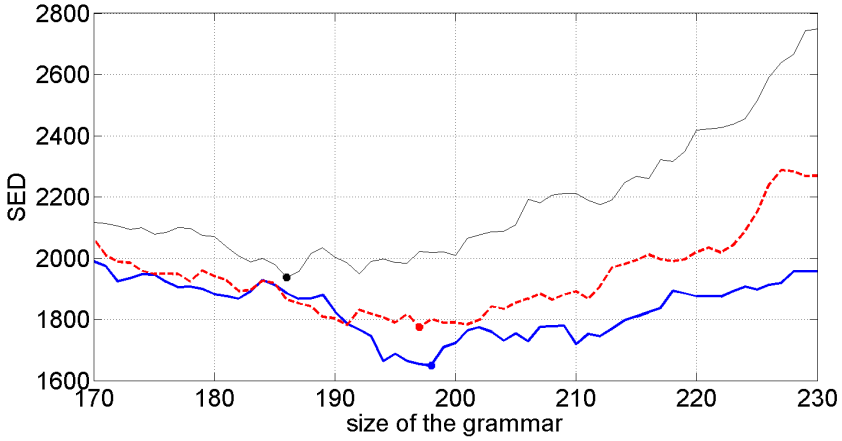


Figure 4.5: *The effect of the grammar size to quality of recognition of the test set. The sum of edit distances (SED) as function of grammar size are plotted for IG, MI and Freq variants with bold solid, dashed and thin solid lines respectively. The optima, marked with solid circles, for each variant are reached as follows: IG: 1649 with grammar size 198, MI: 1774 with grammar size 197, Freq: 1937 with grammar size 186*

the stopping criterion for the inference of CFG. It clearly took the inference too far and started to form too long patterns. That caused them to be too sparse. The sparseness of labels diminishes the training material and causes some of the trained labels to be non-existent in the test set.

In table 4.4 some examples of perfectly recognized sequences, averagely recognized sequences and worst cases are given. The errors in the average examples are results from missing words or letters like *a, the, n* or *blue*. In the worst cases, in the middle of the hypothesized sequences can be seen many patterns that only exist at the beginning of sequences in the training set.

Table 4.4: *Examples of hypothesized sequences from recognition process and corresponding references with their mutual edit distance. These are taken from the best performing variant IG. Three classes recognition results are presented in the table below: perfectly recognized sequences, sequences with average error and the most poorly recognized sequences. Erroneous hypotheses with edit distance 4 or 5 are the most common ones with 50 and 42 occurrences respectively.*

Five examples of perfect recognitions

'yellow'
'whereistheroundapple'
'daddylooksatasmallairplane'
'doyouseethecleantoy'
'hereisabigdirtytruckandatelephone'

Five examples of average recognitions

hypothesis	**reference**	**edit distance**
'daddygivesacleanbottle'	'daddygivesthecleanbottle'	4
'cryingbabyseesaanimal'	'thecryingbabyseesananimal'	4
'doyouseethedoll'	'doyouseethebluedoll'	4
'happywomanlikebottle'	'thehappywomanlikesabottle'	5
'ogdaddylooksatthebottle'	'saddaddylooksatthebottle'	5

Five worst recognitions

hypothesis	**reference**	**edit distance**
'takewhereistheanimal'	'shetakesasquareanimal'	17
'mummycryingcleancar'	'mummylooksatacleancow'	18
'thereisahereisastheediblecarballporsche'	'thereisaroundediblecarandaporsche'	20
'hereihereisadirtydoyouliketruckseesaeagle'	'hereisadirtyedibletruckandaneagle'	22
'hereisabigedibledoyoulikedoyouhaveaneagle'	'hereisabigediblebananaandaneagle'	25

Chapter 5

Conclusions and Discussion

The purpose of this thesis was to develop general methodology for discovering associations between two co-occurring and qualitatively different sequential data streams. A novel method for discovering associations between two co-occurring and qualitatively different sequential data streams (VQ speech and text) was presented. The self-supervised method is based on first discovering patterns with grammatical inference from textual representation and then using them as labels in weakly supervised learning in the vector quantized speech representation. The method was shown to acquire promising predictability of the textual representation when given the corresponding VQ speech signal. The proposed methodology and the results were recognized at Interspeech 2013 conference in peer reviewed conference proceedings [KRL13].

The CM algorithm was selected as the core of the presented self-supervised learning system for two reasons: it has already been proven effective in real keyword discovery from utterances and for the urge to push its limits with the same CAREGIVER research material. The generality of the presented system was higher than the previously introduced studies. This was accomplished by reducing a priori knowledge on the data sequences where the labels were derived from. In contrast to previous studies the labeling was automatic and the knowledge on the order of labels in textual representation was utilized in the segmentation of vector quantized (VQ) speech in the training phase.

Selecting a greedy agglomeration algorithm for discovery of labeling for the weakly supervised learning brought significant restrictions and loss of generality. One of the sequential data streams has to be noiseless, in the sense that all of the instances of the same pattern in the labeling have to be identical.

The associations are found only to one direction since the system learns

only to detect instances of trained labels but not to produce them. In given speech-text context the learning agent is mute but sensing, so to speak.

The presented method works only in batch mode, which can be considered as a weakness. With the current setup a learning agent can recognize only what has been previously introduced to it. There is no built in novelty / anomaly detection. If an instance of a truly novel or a anomalous label would be presented to it in the recognition, it is forced to select something from its learned set of labels as an output/recognition result. As the amount of data and the amount sensors increase in everywhere in both modern business and leisure life, the batch processing is started to be considered as old school. The need for online incremental learning methods is increasing. There is no time and/or capability to both store vast amounts of data and someone to analyze it later.

The nature of the corpus was potentially beneficial, because the vocabulary is statistically balanced. The utterances are grammatically correct but not necessarily sensible or logical. Since there are less word-to-word dependencies, it is more likely that the grammatical inference first discovers the text patterns matching real words and then joins them, rather than discovering text patterns spanning over two incomplete words. Additionally the corpus has the utterances two times. Double utterances have the same content, but different recordings. This fact was not taken into account in any way.

If the presented methodology would have been treated more like an ASR, a statistical language model could have been created for the recognition, which would include estimated conditional probabilities of the text patterns given the previous pattern. The usage of a language model would allow the creation of multiple hypothesized text pattern sequences and selecting the most probable. For example, in the recognition results there are some cases where two text patterns, that both exist only as the first sub-strings of utterances in training, are hypothesized to the beginning of an utterance: *"hereihereisa..."*. At least these kind of errors would be eliminated, since $P(\text{“}hereisa''|\text{“}herei'') = 0$. Creation of a language model was not tried because the domain expertise was kept to minimum and the methodology as general as possible.

Based on the results from the conducted experiments presented in the previous chapter and after the restrictions and shortcomings listed above it is still safe to say, that learning of associations between two qualitatively different data streams is possible, at least up to some level. The presented methodology could work with variety of different other data stream pairs as long as the requirements are met natively or through pre-processing. One of these situations, not far from the presented one, could be phonetic transcrip-

tion in the place of text.

Some sort of real world application for the presented methodology could be in industrial (manufacturing) process. Measurements can be made on the same process but from temporally different phases. In training of the models for patterns, these data streams can be aligned if the mutual delays are known accurately enough. This kind of configuration could help in gaining knowledge on the governing processes of a production system or assist in early fault detection / system health monitoring.

Some domain expertise was used to hand pick the appropriate algorithms to each given sub task. If both of the data streams would be noisy, the algorithms would naturally be different. That situation would also be a natural and intriguing topic of research as a continuation: inferring structure from noisy data stream and using that as labeling in weakly supervised learning for the other noisy stream of data.

There are a few other things that would be interesting and worth trying in the presented learning problem with noiseless-noisy signal pairs. For example, DTW might be helpful in improving the extraction of VQ speech segment corresponding to given label. The usage of DTW could work especially in the presented batch mode, where all the training data is stored as it is. Also, an interesting topic of discussion is using greedy agglomeration for the VQ speech signal. It is not really intuitive whether agglomeration of VQ speech leads to feasible results, as the agglomeration produces new indices, that are sparser, but more definitive. Also the CM matrices would become larger and sparser.

Bibliography

[Aim09] Guillaume Aimetti. Modelling early language acquisition skills: Towards a general statistical learning mechanism. In *Proceedings of the Student Research Workshop at EACL 2009*, pages 1–9, Athens, Greece, April 2009. Association for Computational Linguistics.

[AtBA+10] Toomas Altosaar, Louis ten Bosch, Guillaume Aimetti, Christos Koniaris, Kris Demuynck, and Henk van den Heuvel. A speech corpus for modeling language acquisition: Caregiver. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may 2010. European Language Resources Association (ELRA).

[Bel72] Richard Bellman. *Dynamic programming*. Princeton University Press, Princeton, 1972.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[BL97] Avrim L. Blum and Pat Langley. Selection of relevant features and examples in machine learning. *ARTIFICIAL INTELLIGENCE*, 97:245–271, 1997.

[BPPM93] Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Comput. Linguist.*, 19(2):263–311, June 1993.

[BPSW70] Leonard E. Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occurring in the statistical

analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41(1):pp. 164–171, 1970.

[BY03] Dana H. Ballard and Chen Yu. A multimodal learning interface for word acquisition. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP03)*, pages 2–5, 2003.

[Chi06] David Chiang. An introduction to synchronous grammars, 2006.

[CK10] Nitin S. Choubey and Madan U. Kharat. Grammar induction strategy using genetic algorithm: Case study of fifteen toy languages. *The Pacific Journal of Science and Technology*, 11(1):294–300, 2010.

[CL06] Ronald R. Coifman and Stéphane Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30, July 2006.

[DHS00] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[DM80] Steven Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 28(4):357–366, 1980.

[For73] G. David Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[GFK05] Todor Ganchev, Nikos Fakotakis, and George Kokkinakis. Comparative evaluation of various MFCC implementations on the speaker verification task. *Proceedings of the SPECOM*, 1(3):191–194, 2005.

[Gib86] James J. Gibson. *The Ecological approach to visual perception*. Lawrence Erlbaum Associates, September 1986.

[Goo12] Google translator, 2012. `http://translate.google.com/about`.

[Hot33] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441, 1933.

[hSGC+14] Man hung Siu, Herbert Gish, Arthur Chan, William Belfield, and Steve Lowe. Unsupervised training of an hmm-based self-organizing unit recognizer with applications to topic classification and keyword discovery. *Computer Speech & Language*, 28(1):210 – 223, 2014.

[Koe05] Philipp Koehn. *Europarl: A parallel corpus for statistical machine translation*, volume 11. 2005.

[Koh97] Teuvo Kohonen. *Self-organizing maps*. Springer, 1997.

[KRL13] Juho Knuuttila, Okko Räsänen, and Unto K. Laine. Automatic self-supervised learning of associations between speech and text. In *Proceedings of ISCA Interspeech*, pages 465–469. ISCA, 2013.

[Lai11] Unto K. Laine. Entropy-Rate Driven Inference of Stochastic Grammars. In *Proceedings of ISCA Interspeech*, number August, pages 2489–2492, 2011.

[LBP09] Annick Lesne, Jean-Luc Blanc, and Laurent Pezard. Entropy estimation of very short symbolic sequences. *Physical Review E*, 79(4):1–10, April 2009.

[Leo84] R. Gary Leonard. A database for speaker-independent digit recognition. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '84.*, volume 9, pages 328 – 331, mar 1984.

[Lev65] Vladimir I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Doklady Akademii Nauk SSSR*, 163(4):845–848, 1965.

[Lop08] Adam Lopez. Statistical machine translation. *ACM Computing Surveys*, 40(3):1–49, August 2008.

[LS01] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems*, pages 556–562. MIT Press, 2001.

[MÖ7] Meinard Müller. *Information Retrieval for Music and Motion*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[Mac67] J MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, 233(233):281–297, 1967.

[MD09] Kaspar Meyer and Antonio Damasio. Convergence and divergence in a neural architecture for recognition and memory. *Trends in neurosciences*, 32(7):376–82, Jul 2009.

[Mic12] Microsoft translator, 2012. `http://www.bing.com/translator`.

[New04] Mark E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6):066133+, June 2004.

[Oat02] Tim Oates. Peruse: An unsupervised algorithm for finding recurring patterns in time series. In *In Proceedings of the IEEE International Conference on Data Mining*, pages 330–337, 2002.

[PG08] Alex S. Park and James R. Glass. Unsupervised pattern discovery in speech. *Ieee Transactions On Audio Speech And Language Processing*, 16(1):186–197, 2008.

[Rä10] Okko Johannes Räsänen. Fully Unsupervised Word Learning from Continuous Speech Using Transitional Probabilities of Atomic Acoustic Events. In *Proceedings of ISCA INTERSPEECH*, pages 2922–2925, Chiba, Japan, 2010.

[Rä11] Okko Räsänen. A computational model of word segmentation from continuous speech using transitional probabilities of atomic acoustic events. *Cognition*, 120(2):149–76, August 2011.

[Ris78] Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465 – 471, 1978.

[RJ93] L. Rabiner and B. Juang. *Fundamentals of Speech Recognition*. Prentice-Hall Signal Processing Series, Englewood Cliffs, New Jersey, 1993.

[RK89] Helge Ritter and Teuvo Kohonen. Self-organizing semantic maps. *Biological cybernetics*, 254:241–254, 1989.

[RL12] Okko Räsänen and Unto K. Laine. A method for noise-robust context-aware pattern discovery and recognition from categorical sequences. *Pattern Recognition*, 45(1):606–616, January 2012.

[RLA09] Okko Johannes Räsänen, Unto Kalervo Laine, and Toomas Altosaar. A noise robust method for pattern discovery in quantized time series: the concept matrix approach. In *Proceedings of ISCA Interspeech*, pages 3035–3038. ISCA, 2009.

[RS07] Lawrence R. Rabiner and Ronald W. Schafer. Introduction to digital speech processing. *Foundations and Trends® in Signal Processing*, 1(1–2):1–194, 2007.

[RSL10] Barry Ridge, Danijel Skocaj, and Ales Leonardis. Self-supervised cross-modal online learning of basic object affordances for developmental robotic systems. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 5047 –5054, may 2010.

[Sch92] Robert Schalkoff. *Pattern recognition : statistical, structural and neural approaches*. Wiley, Singapore, 1992.

[Sch07] Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, Aug 2007.

[SDVh08] Veronique Stouten, Kris Demuynck, and Hugo Van hamme. Discovering phone patterns in spoken utterances by non-negative matrix factorization. *IEEE Signal Processing Letters*, 15:131–134, 2008.

[SK83] David Sankoff and Joseph B. Kruskal. *Time warps, string edits, and macromolecules: The theory and practice of sequence comparison*. Addison-Wesley Publishing Company, Inc., 1983.

[Sol64] Ray J. Solomonoff. A formal theory of inductive inference. part II. *Information and Control*, 7(2):224–254, June 1964.

[SW96] Yoram Singer and Manfred K. Warmuth. Training algorithms for hidden markov models using entropy based distance functions. In *Advances in Neural Information Processing Systems 9*, pages 641–647. MIT Press, 1996.

[tBC07] Louis ten Bosch and Bert Cranen. *A computational model for unsupervised word discovery.*, page 1–4. 2007.

[Van08] Hugo Van Hamme. HAC-models: a Novel Approach to Continuous Speech Recognition. In *Proceedings of ISCA Interspeech*, pages 2554–2557, 2008.

[VH08] Hugo Van Hamme. Integration of asynchronous knowledge sources in a novel speech recognition framework . In *ISCA ITRW workshop on Speech Analysis and Processing for Knowledge Discovery*, 2008.

[WF74] Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *J. ACM*, 21(1):168–173, January 1974.

[WMS01] Chin-Chung Wong, Helen M. Meng, and Kai-Chung Siu. Learning strategies in a grammar induction framework. In *Proceedings of the Sixth Natural Language Processing Pacific Rim Symposium, November 27-30, 2001, Hitotsubashi Memorial Hall, National Center of Sciences, Tokyo, Japan*, pages 153–157, 2001.

[WS93] Ming-Wen Wu and Keh-Yin Su. Corpus-based Automatic Compound Extraction with Mutual Information and Relative Frequency Count. *Proceedings of ROCLING VI*, 1993.

[Xie05] Lexing Xie. *Unsupervised pattern discovery for multimedia sequences*. Doctoral thesis, Columbia University, 2005.

[YBA05] Chen Yu, Dana H Ballard, and Richard N Aslin. The role of embodied intention in early lexical acquisition. *Cognitive science*, 29(6):961–1005, Nov 2005.